UNIVERSITÀ DEGLI STUDI DI GENOVA

DIME - Dipartimento di Ingegneria Meccanica, Energetica, Gestionale e dei Trasporti

DOCTORAL THESIS

Corso di Dottorato di Ricerca in Ingegneria Meccanica, Energetica e Gestionale
Curriculum in Economia e Gestione

XXXI ciclo

AGENT-BASED MODEL AND SIMULATIONS OF THE MANAGEMENT OF PORTS: THE IMPORT PROCESSES AT THE PORT OF GENOVA

Tutor:	Candidate:
Prof. Silvano Cincotti	*Dott. Onur Senturk*

CONTENTS

List of Figures

List of Illustrations

List of Tables

Abstract

This thesis addressed to use of Agent-Based Modelling (ABM) for the development and implementation of the import process of goods in a port that is suitably applied to help, plan, structure the development of the port model. The main goal of the modelling and implementation was to reach to fruition as more organized, fast and efficient complex logistics network, through development policies. To this purpose, I developed an agent-based model (ABM) of a port that is populated by the real main actors (stakeholders) whose are involved in the port activities such as maritime, customs, financial police etc. The model of the port simulates the actual port processes, i.e. acceptance of the goods, sending them, controlling of the legality, or not to import goods, the transportation planning etc. Agent-based models (ABMs) are being used in modelling in economies as complex systems that is a relatively recent approach in economies [62]. It has increasingly been attracting many scholars belonging to several sub-fields, becoming both a complement and a substitute for more traditional economic-modeling methodologies [62]. We can mention that ABMs are considered as a valid and effective competitor of standard Dynamic Stochastic General Equilibrium (DSGE) models in macroeconomics [26]. The main advantages of using ABMs arrive two main additional values if we compare it with its equivalent systems. ABM provides more descriptive richness, as they characterize ecologies of agents, locally interacting through non-obvious network structures, learning using incomplete information, and competing within imperfect markets. Second, the modeler developing an ABM has typically more flexibility in both input and output validation of its model [34].

Ports have an integral role of our economy, they are strategic places of exchange, and especially over the last few decades and with the phenomenon of globalization, the ports are a reality in continuous movement and growth. Therefore, they are operating places of extreme complexity, especially in their logistics functions of transport management.

The thesis discusses the business process is implemented for developing a computer supported management tool to handle the port activities flow. The tool is designed for the integration in a virtual infrastructure that allows an advanced operational management of port traffics. By modelling the time documentation according to the specification of the Genoa case, the business case of the port of Genoa is tested.

Results show that the mechanism implemented simulates the actual process. Moreover some bottleneck are discovered, such as delays to the handling of the containers and queues formation due to missing documentation or documentation with errors or not ready.

Introduction

The thesis encloses the results that is achieved during the last three years of PHD program in mechanical, energy, and management engineering and agent-based modelling and simulations are focused as the main field. I discussed the use of Agent-based modelling for development of the import process of goods, implementing and simulating them. Therefore, I aimed to make more organized, fast and efficient complex logistic network through ad-hoc development policies.

Ports have a big, important and integral role of the economy. They are strategic places of exchange, and especially over the last few decades and with the phenomenon of globalization, the ports are a reality in continuous movement and growth. We have to mention that also they are operating places of extreme complexity, especially in their logistics functions of transport management. Accordingly this purpose, I develop an agent-based model (ABM) of a port that is populated by the real main actors (stakeholders) involved in the port activities. The model simulates the actual port processes, i.e., the sending of goods, the acceptation or not of imported goods, the planning of transports etc. With this framework, the business process is implemented for developing a computer supported management tool to handle the port activities flow. The tool is designed for the integration in a virtual infrastructure that allows an advanced operational management of port traffics. By modelling the time documentation according to the specification of the Genoa case, the business case of the port of Genoa is tested. The main goal of the research was to make research on the port's complexity of the logistic functions of the transport management.

Business process modeling is usually conducted by business process analysts to capture business requirements, enable a better understanding of business processes, and facilitate communication between business process analysts and IT experts, who are often involved with the scope to monitor automate process activities [49].

Designing and implementation of a new model is really complex, consumes so much time, and is error-prone. It requires extensive knowledge to understand company's operation and business rules perfectly. Because of this developing efficient business process becomes more and more important.

Business processes are a collection of one or more activities performed following a predefined order to achieve an objective business goal, usually within the context of an organizational structure that defines functional roles or relationships. Furthermore, a process can be regarded as work activities organized in a specific order in time and space, with clearly identified inputs and outputs [23]. Briefly, processes can be addressed as a set of activities that connects each other with inputs and outputs.

The Business Process methodology to collect the data for modelling business processes has six stages [64], [20], [57], [5], [93]. According to the Business Process Management, the six stages are process identification, process discovery, process analysis, process redesign, process implementation, and process monitoring and controlling. In these six stages, process modelling is an important part of the process discovery stage. Many different methods and notations have been proposed to support these activities, such as object-oriented approaches based on UML class diagrams, and entity-relationships models [76], [24]. Several proposals are based on BPMN or UML activity diagrams [87], [75], [3]. Recently, the agent-based models and simulations (ABMSs) have been employed in business processes modelling [28]. Indeed, agent-based models are increasingly being used to study and examine several transportation issues, which range from traffic flow to air traffic control [83]. According to [27] and others, the ABM approach seems very promising for simulating stakeholder interactions such as in a seaport environment. In an ABM, different agents have different roles and also individual goals. The use of agents representing the different actors permits to compare several solutions for making the best use of the resources in the total port operations process. In this paper we pursue and extend this trend, by focusing on an entirely topic: planning and management of ports. The execution of the ABM allows to investigate patterns that are interesting for analysis. Emerging behaviors of the several agents modeled at a micro level and then simulated on a macro level permit to understand the complex interactions of the modeled agents. This understanding contributes to a more structured approach on stakeholder relations management. Furthermore ABM allows the entities to communicate and allow researchers to study the behavior under complexity. In particular, we build an agent-based model of the port of Genoa, using its futures in structural terms.

Agent-based modelling (ABM), already employed for the study of complex systems, such as financial markets [2], [69], [70], economic systems [71], [72], [73], [6], [34] and materials' properties [68], is an alternative approach able to address processes with a large number of units.

In this paper, the agent-based model and simulations of the import process of goods is presented. As a reference case the port of Genoa has been considered. The import processes of goods has been modelled considering the agents' behavior and the documentation necessary. In particular, for modelling the import process of goods the following agents have been defined: the Agent Maritime, the Customs, the Custom Clearance Agent, the Delivery Service, the Deliverer, the Terminal, the Finance Police, the Gate and the Terminal Area. The agents communicate among each other by means of messages and simulating the real processes, for example sending the goods, accepting or rejecting them, planning the transport. The agent-based tool chosen is Flexible Large-scale Agent Modelling Environment (FLAME) that is a high-level modelling language. The FLAME provides a tool to develop agent-based models that can be run on High Performance computers. Models are created based upon a model of computation called (extended finite) state machines.

Research Questions and Objectives

The problems that are expressed above, four main research questions are extracted.

First research question is 'How to design an appropriate model of the import processes?

Second question is 'Which direction can be used for the implementation of the designed model?'

Third question is 'How to make more organized, fast and efficient complex logistics network in the port?'

Fourth question is 'What are the parameters and functions that are contained in the model?'

In order to address the research questions, an agent-based model was designed and developed and a set of simulations with different parameters was performed and analyzed. All these parameters are explained in the 'Model Construction' and 'Development Process' chapters.

The ports have a huge potential of growth thanks to globalization. They are an exchange places for the goods all over the world and they have an important role for the economy. It is worth saying that logistics functions that are mentioned in the modelling chapter, have extreme complexity. So, the thesis addressed to model and implement of the import process of the goods to achieve well organized, fluent, effective and efficient complex logistic networks. To obtain these objectives, the model is created and it is filled up with sub-models (agents), variables, environments same as real actors in the port activities. These port activities are performed by the functions inside the model. Real actors are represented as the agents. The main objective is to simulate the actual port processes of the import such as sending goods, acceptation procedures, clearance, confirmation of the transportation, planning of transports and their documentation procedures etc. Due to these reasons, business process is implemented to develop a computer supported management tool to handle the port activities flow. The tool is designed for the integration in a virtual infrastructure that allows an advanced operational management of port traffics. The business case is tested to model the time documentation in the Genoa case.

The structure of the thesis

The thesis consists three main parts and five chapters.

Firstly, theoretical introduction explained as Agent-Based Modelling and Import processes, it is performed through research on the literature and Flexible Large-scale Agent Modelling Environment (FLAME) addressed as the tool that is designed for the integration and implementation of the port model. Simulations are executed by the FLAME for reality of the import process.

Secondly, Design and implementation of the Agent-Based Model are examined in details. In order to have a reality of the simulations, all the import processes are created as functions of the model. All the activities and documents are set up as variables and environment. In addition real actors are represented by agents.

Lastly, analysis of the data returned by the simulator (Xparser on Flame) that modeled, and the parameters are highlighted. Flame contains Xparser program which analyzes the model in XML and transforms it into the source code of the simulation program, compiling it together with the implementation codes of the functions of the agents and the template files.

Chapters of the thesis are organized as; Section I presents the General model design, Section II is the development and execution of the model, Section III shows the literature and Section IV is the Model. Finally, Section V provides the results and conclusion of the study.

Section I addresses theoretical part of the Agent Based Modeling. It helps to answer the advantages and disadvantages of the Agent based approach. In addition, modelling techniques such as analytical methods and the simulations techniques and also the Monte Carlo Method are addressed.

Section II contains development process of the model that is created. Agents, variables, elements, functions are explained in detailed. Execution of the model on FLAME is presented. Stategraph of the model are explained on the basis of each agents individually. Launching simulations and reading results processes are explained. Moreover, Virtual appliance are mentioned. The working principle of the Virtual Machine clarified. Furthermore, the code and debugging procedures are written. The range of the parameters are decided in this chapter.

Section III includes the literature of the port. The types of the ports, physical structure, shipping procedures, and a brief summary of the Genova Port occurred.

Section IV, the model description of import processes of the port is presented. It is built the model using the features of the Genoa port in structural terms. It is considered the stakeholders of the Genoa port and their mutual interactions, using the real network and the real data related to the processing of the documents exchanged among various actors (agents) included in the model. The roles of the agents and their interactions inside the model are explained.

Section V covers the conclusion of the study and the results.

Chapter 1

Model Design

1.1. Baseline of the model - Agent-Based Model

A descriptive or predictive computational model underlies the computer modelling and simulation. It has the scope to investigate the behavior of systems which it depicts and to evaluate strategies in order to improve its functioning.

Simulations allow us to evaluate and predict the dynamic series of events or processes which follow the imposition of certain conditions by users.

As it is mentioned on the introduction part, the Agent Based Model is used to produce port transactions for our simulator. The typology is chosen after carefully analyzed the various pros and cons (like obtainable results or the difficulties of the development of the model etc.). The most important examples are created by the research group that is coordinated by Prof. Silvano Cincotti. Developed models are in the financial economic field, such as the Genova Artificial Stock Market (GASM) and EURACE [82], [11] [15].

The “Model” is an abstract and simplified representation of an assigned reality and is it used to examine a phenomena or to forecast it. The “Simulation Model” is the manifestation of a model, by means of a computer program that includes the computing algorithms and mathematical equations which describe the behavior and performance of a system in the real world scenarios.

Agent Based Modelling and Simulation (ABMS) belongs to a category of computational models that use the dynamic actions, reactions and intercommunication protocols among the agents in a shared environment, to assess their performance and understand their behavior and properties.

The ABMs or multi-agent systems (MASs) consist of a set of elements (agents), characterized by some attributes, which interact with each other in a given environment through the definition of appropriate rules. [27].

A software agent is a software abstraction as the programming specifications that is objects, methods, procedures and functions but it hasn’t attributes and doesn’t work using logic-based methods. Unlike an object, an agent is defined by its predetermined actions.

There are a minimum set of features to describe a software agent. ‘‘A software agent is autonomous; capable of operating as a standalone process and performing actions without user intervention’’ [56].

According to the BDI (Belief-Desire-Intention) framework an agent has beliefs about its environment and itself, computational states (desires) to keep and computational states (intentions) to realize.

ABMS is commonly exploited in scientific disciplines such as economics, biology, ecology, sociology, social sciences and many other disciplines of STEM (Science, Technology, Engineering and Mathematics) to reproduce and study dynamic large-scale complicated systems and to learn new behaviors.

Complex systems can be seen as sets of agents or entities which have specific features and interact among each other to establish relationships and encourage automated reasoning and problem-solving.

Mainly, ABMS tools help the scholars to understand the macroscopic behavior of a system starting from the micro-level properties, constraints and rules. Specific states and sets of functional attributes, properties, or rules can be assigned to agents using predefined parameters which induce special actions.

Computational models, especially Agent-based Models are acceptable as new addition analytical toolkit that is used in the social science, particularly in economics, finance and marketing. They allow a new level of analytical analysis in between classical modelling approaches that describe an individual’s behavior and decision making process, or the relationships between aggregated measures that characterize a population as a whole; the tools used in these traditional approaches are usually statistical models, such as regression or structural equation model or mathematical methods, including differential equations and analysis of equilibria and limiting behavior.

Agent based models and related computational tools do not replace these models, but add a new dimension to the range of issues can address with formal means: with ABM it is possible to formalize how individual actions and decision making bring about aggregate characteristics of a population, which is especially interesting when the aggregate is brought about through the individual interactions. ABM also takes into account that individuals are generally not isolated but are interdependent and interact with each other through actions and interactions of different types, for example directly and indirectly, intentionally or unintentionally [42].

In social systems, the study of a political or economic system requires the understanding of the mechanisms underlying the interaction of individuals which assemble the system, indeed the results don't represent the mere sum of parts.

Computational models generally have the advantage of being monitored in great detail and are susceptible to a wide variety of manipulations. This is particularly useful in the study of social systems, as already presented, because in these manipulations and experiments are often impossible in the original system, due to practical, ethical or technological limitations [42], and for this reason agent models are therefore recommended and used more and more frequently as promising heuristic techniques to solve problems whose domains are distributed, complex and heterogeneous [37].

This type of model specifies the rules how the state of the system changes step by step, while the computer calculates the development of the system over time: in essence we use the computer to obtain numerical solutions to very complex and non-linear mathematical problems [42].

Agent-based computing has been suggested as a promising technique for the problem whose domains are distributed, complex and heterogeneous [84], [88]. Parunak [67] suggested a first formalization of a set of resources allocation problems using agent-based approaches.

Due to the fact that ABM models whole systems, not individual hypotheses, the development process of a systems theory has a peculiar structure consist of a core of theoretical assumptions and a collection of auxiliary hypotheses (Cioffi-Revilla, 2010). If a computational model fails to reproduce empirical patterns, the first conclusion to be drawn from this is that the set of assumptions as a whole is inconsistent with the observations, but instead of discarding the whole model/theory, it would be better to use a systematic analysis of model components to improve the model performance. This process is different to conventions for handling statistical models, since these usually test a well-defined hypothesis, not a set of assumptions and combination of hypotheses [42].

Additionally, agent-based modelling can be used to provide pragmatic solutions to many problems related to environment, healthcare and finance. Moreover, these models have long been applied to epidemiological problems and have contributed to improve the welfare of humanity.

Other application areas of ABMS are: design of self-organizing systems either continuous or discrete-event, simulating fluid flow-rates, immunology, physiological fluctuations including the systems' ability to react to a trail of environmental impulses/stimuli, transportation and logistics, failures detection and diagnosis in distributed systems, manufacturing, production, and so on.

It was the prospect of modeling the interactions of many autonomous individuals that greatly increased the interest in simulation as a tool for social sciences [42]. In these models the overall behavior, generally very complex, does not develop through central coordination but, through a bottom-up mechanism, arises from the actions and interactions of the actors at the micro level [63].

Therefore, about last point, it is worth defining that the main actor of this model, an agent consists only a set of logical rules of behavior and a list of internal states, representing, for example, its memory, the mood or capabilities. All of the agent's combinations collect information about his current situation, both internal and external, and then match this input with the set of rules infer his reaction to the situation. [42].

To understand better of social systems, it is important to say that all populations of the agents can be social. Because they interact and influence each other when they are combined in a computer simulation. So it is possible to study deeply the development of their interactions in time.

Using agent-based modelling, users can establish interactions between the agents in their environment, then to develop their own real-world system models. An Integrated Development Environment (IDE) is a standalone application programming environment that includes a typical code editor, compiler, tester/debugger and visualizer or interactive Graphical User Interface (GUI) builder.

These models permit to visualize on the user interfaces the agents' overall behavior following their micro-level actions. These actions have direct effects on the macro-scale properties of the entire unit. Therefore, the dynamical interactions among the agents, the changes on each agent due to these interactions and the effect on the overall system performance can be visualized on the user interfaces.

Over the years, several agent-based modelling and simulation tools have been made. Each of them has a specific programming syntax and semantics for the agents and has a different generality, usability, modifiability, scalability and performance.

The most important characteristic of an ABMS tool is its goal; that is the domain in which it can carry out modelling and simulation.

The comparison among the agent-based modelling and simulation software tools has to take into account the following aspects:

- source-code specification (open or closed source), online availability, distribution license / legibility;
- interaction mechanism of agents during simulation process;
- programming language requirements and availability of an Application Programming Interface (API) and built-in libraries for developing agent-based models;
- availability of a graphical programming interface for realizing code and simulation, or immediate visual programming user interface for reading the model results;
- characteristics of the compiler, operating system support, and platform/hardware requirements/constraints for the model realization;
- effort/input employed for the design and development of users' model;
- computational modelling strength or scalability of the models developed related to particular toolkits;
- assessment of the ABMS tools with respect to coverage of satisfied application fields;
- agent-based modelling and simulation functionality provided by each mentioned ABMS tool.

ABMS toolkits often require the knowledge of the programming languages, such as: C/C++, Python, Java, Smalltalk, Basic; but there are commercially available toolkits with ready-to-use Application Programming Interfaces (APIs), add-ons and libraries.

Software agent-based simulation and modelling can work on notebooks up to high performance computers. Computer programming languages with general end or dedicated toolkits can be used for developing ABMS.

Generally, the simulations of agent-based models (ABM) expect the processing of millions of agents so it is frequently required to work on high performance parallel programming platform in order to design distributed simulations and to shorten the simulation time.

Whole models include all known expressions of a system, unlike of partial models which consider a set of classic pre-assumptions related to the most significant characteristics. In that case, a wide range of experimental simulations can be carried out, undergoing tests the models with several hypothesis.

Agents generally have limited abilities, especially local ones, to perceive their environment. They can be heterogeneous in terms of their abilities or objectives and they can have means of interaction and communication.

Wooldridge & Jennings (1995) outline the typical properties of IT agents [10]:

- Autonomy

Agents monitor their actions and their internal state. In particular the user does not interfere with the decision-making process after having specified their own rules).

- Social Ability

  Interacting with other agents, based on a language or common actions, through specific functions.

- Reactivity

  Agents are able to perceive their environment, including other agents, and they are able to react on the basis of these perceptions.

- Proactivity

  In addition to the only reactions to their environment, they are also able to undertake initiatives, engaging in directed behavior.

These properties can be managed by the cognitive model of the agent. There are different types to manage. One of the most classical model is the BDI (Beliefs-Desires-Intentions) model that considers a part of the beliefs of the agent on the environment in where it is located and operated that is the results of its knowledge and perceptions, on the other side objectives (Desires) that can be seen as computational states. Intersection of these two are obtained as a new intentions that can be translated directly into actions and also can be defined as computational states [74].

Furthermore, the description of the agents are not enough to create an ABM, the development of an Agent-based Model requires a complete description for a set of basic elements/building blocks, it can be derived from Billari (2006) and Weiss (1999).

- *The object of the simulation.* It has to be clarified what is the phenomenon / problem to be reproduced, defining the space in which the simulation takes place.
- *The agents' population.* The agents can be grouped in different categories with common characteristics that reproduce the various components of the system.
- *The adaptive capability of each agent category.* How the agents of each category behave in certain situations, in another words, Agents of each category present a specific adaptive capability, namely the degree of reactivity and pro-activity.
- *The interaction paradigm of agents.* Each agent can interact with the agents of the same or other categories and in this context, agents can competitive, cooperative or negotiable if they have conflictive objectives. (If they have not, agents can not be competitive.)

It is worth talking that different interaction paradigms have been defined in the literature by the Weiss (1999) that can be mainly classified as; first is *cooperative paradigms*, when there is a partially (half) coordination between uncompetitive agents, and second is *competitive paradigms* that some rules are defined for the behavior of the competent agents.

The agent models emphasize the importance of the agents' interactions to explore how social phenomena generate, these phenomena are realized in real life, as Epstein and Axtell (1996) have already observed, the simple entities, which interact with simple local rules, can produce very complicated behaviors, as previously noted.

The models must necessarily include relevant aspects of the agent environment to provide context of interactions. The environment can be physical or abstract, such as a geographical landscape or a social network, and may also contain passive agents, such as objects or resources that interact with active agents [42].

A characterization for an ideal application of the ABM technology was produced by Parunak (1999) and follows [21], which connects with the basic characteristics previously explained for the general simulation models.

Firstly, it must be *modular*. In the sense that each entity has a well-defined set of state variables that is distinct from those of its environment and that the interface to the environment can be clearly identified. Secondly, it must be *decentralized*. It means that the application can be decomposed into autonomous software processes that are able to perform useful activities without a continuous direction from another software process. The third, it must be *changeable*. It means modifiable. In the sense that the structure of the application can change quickly and frequently. The fourth is ill-structured. In another words poorly structured. It means that all information about the application is not available when the system is being designed. The fifth and the last one, it must be *complex*. In the sense that the system presents a large number of different behaviors that can interact in sophisticated ways.

Some ABMS packages (Altreva Adaptive Modeler, AgentSheets, Envision, ExtendSim, Framsticks, Mathmatica (Wolfram), MASS, MASyV, Mimosa, PedSim, Repast Simphony, SimEvents (MATLAB), SOARS, StarLogo, Sugarscape, VisualBots, Simio, Simul8) have visual easily understood graphical tools which permit to realize the model using flexible drag-and-drop interfaces. İn addition they have real-time visualization with chart and plot to study the models and their properties.

In the past, the programming languages normally used were: C, Smalltalk and Java, nowadays the ABMS software tools (Ascape, Breve, FlexSim, GAMA, GridABM, GROWLab, JAMEL, JAS, LSD, MASS, MASON, MASyV, NetLogo, PedSim, PS-I, Repast- J or Repast-3, Repast Simphony, Swarm, UrbanSim) are conceptual frameworks with standard APIs, predesigned agent-templates and embedded libraries of procedures, which can be included in the custom formed programs. The developers build simulation models by means of calls, then re-calling built-in functions within the modelling toolkit.

Sets of packages, put together within the common standardized graphical Integrated Development Environments (IDEs), facilitate the creation of interactive simulation models by means of visual programming IDEs. These are: Altreva Adaptive Modeler, AgentSheets, AnyLogic, FLAME, Framsticks, Mimosa, PedSim, Repast Simphony, SeSAm, StarLogo, and Sugarscape.

From the point of view of application areas satisfied by the ABMS tools, there are several tools which permit the modelling and simulation in social, natural, and human sciences; for example, dynamic computational systems, business, marketing, economics, ecology, traffic situations, the education of Physics, Chemistry, Biology and other natural sciences and so on.

Among the ABMS hands-on instruments including simple easy-to-use interfaces, it is possible to quote: AgentScript, AgentSheets, BehaviourComposer, FLAME, Framsticks, JAS-mine, MIMOSE, MOBIDYC, NetLogo, Scratch, SeSAm, SimSketch, StarLogo, StarLogo TNG, Sugarscape, VisualBots, VSEit. However these systems are useful to implement less complex models.

ABMS software, like, Agent Cell, BSim, Breve, DigiHive, Echo, EcoLab, FLAME GPU, Grid-ABM, JCASim, jES, MASyV, MOBIDYC, Pandora, Swarm, Xholon, covering high-level domains, allow to develop sophisticated models with numerous agents and complex rules.

That is why these simulations process a large number of agents and, memory and storage matters, need to be modelled on the workstations and high-end scientific computing systems. Among these, there are MATSIM, PDES-MAS, Repast HPC and Swarm [1].

How we said above, the Genoa Port Model was implemented using the Flexible Large-scale Agent Modelling Environment (FLAME) developed by Simon Coakley, Mike Holcombe, and others at the University of Sheffield [35].

From the point of view of application domains deal with ABMS tools, we can note that FLAME represents an ideal solution when the ABMS scope lies in the wide categories of Social & natural sciences, Dynamic computational Systems, Business, Marketing, Economics, Planning & Scheduling, Enterprise and organizational behavior and Traffic Situations.

1.2. System Modelling & Simulation

1.2.1. Introduction to Simulation

The main idea of the Simulations, helps to evaluation, prediction and interpretation of the data that is collected with the conditions from the FLAME model database.

Simulations are necessary and widely used by the researchers in lots of fields. They are an experimental tool for making analysis. The most important thing is to help us to use potential computing of data. On the other hand, the difficulty or impossibility of the reproduce of the actual conditions. Because of this reasons, the simulation of the production and logistics processes becomes very important. The systems are characterized with high level of complexity (i.e. high number of presented entities), numerous interrelations between different processes, presence of multiple problems (controllable or not), stochasticity of the system parameters. At that point, in the case of stochastic systems, the functioning of the system is simulated with using probability of distributions to generate system events randomly. After that, statistical observations on the performance of the system will be obtained. In order to this, Simulation model should be developed well. It must describe the operations of the system and how the operations must be simulated. So, the most important part is to build a model that must be complex enough to meet requirements of the case.

Naturally, in order for this to be realized it is necessary to construct a simulation model that allows to describe the operations of a system and how they must be simulated: it is therefore of primary importance to build the model, which must be complex enough to respond to the needs of the case [80].

There are two types of modeling techniques that is used. First is analytical methods and the second is simulation techniques.

Analytical model is a set of equations describing the performance of a computer system. In the analytical model, the components of the system are represented by variables and parameters. Analytical models are constructed and used by capacity planners to predict computing resource requirements related to workload behavior, content, and volume changes, and to measure effects of hardware and software changes. Developing the analytical model provides the capacity planner with an opportunity to study and understand the various behavior patterns of work and hardware that currently exist. Certain factors must be taken into consideration to avoid common errors in model construction, analysis, and predictions. In practical terms, it describes a collection of measured and calculated behaviors of different elements over a finite period of time within the computer system – workloads, hardware, software, and the CPU itself, and can even include the actions and behaviors of its users and support personnel [9].

Simulation is perhaps the best tool used for any non-trivial, real world system. For analysis of complex systems, simulation is often used prior to the operation of the real world system as a mediator for a dynamic situation.

A simulation model produces the dynamic behavior of the system. The components and interactions are represented in terms of functional relationship. During evaluation of the system, simulation model requires the execution (run) of the simulation program (or simulator) that represents the temporal evolution of the system.

The usage of the model brings some advantages and disadvantages [19]. The advantages are;

Increasing of knowledge is the first important point. A model helps to organize theoretical knowledge and empirical observations on the system, lead to understand of the system better, it is worth saying that during the abstraction process, it is necessary to identify which are the components and the interactions relevant to the scope of the study.

System analysis; the usage of model facilitates system analysis.

Modifiability is the capability of being modified. It is more modifiable and manipulable respect to the system that allows to evaluate of various alternatives, compatibility with the definition and the abstraction that is adopted.

Different objectives of the study, the usage of different models helps to evaluate different objectives of the same system.

The limits and disadvantages of the modeling techniques are:

Selection of the model, the selection process of the appropriate abstraction level is not easy and the usage of an inappropriate model can lead to errors of the evaluation [4].

The usage of incorrect model, the risk of using expired (past) model beyond its field or the validity, if the hypothesis and assumptions are not verified anymore, Improper usage of the model due to the extrapolation of the results beyond its applicability because the results can not be considered correctly in this case [4].

Furthermore, the simulation provides indications on the behavior of the system, but not exact "answers" [80].

Simulation models include various types of elements: state variables, events, entities and their attributes, resources, activities and delays [80].

The state variables are the set of variables that describe the system at a given time. These are different variables depends on the types of the system described, because a discrete systems require discrete variables while continues systems require continuous. But it is worth saying that this is not a fixed rule. It is possible that continuous models use discrete variables such as the position of a train can be defined either through the position received from a GPS (continuous) or through observation of the presence or absence of trains on a certain route (train is equal to 1, the train is no there is equal to 0) without specifying the right km. But state variables are distinguished as endogenous variables if their change is due to activities inside the system, and exogenous variables if they are influenced by the external environment of the system. A system is called 'closed' if its' behavior is completely determined by the internal activities, i.e. if there is no exogenous variables, it is called open, if it interact with the external environment, as expressed by the exogenous variables [4].

All the events are instantaneous that change the value of at least one of the state variables. For example the arrival (user) to the system queue, as well as the completion of a service. So related to the variables, there are external events in the system (exogenous events) and internal events (endogenous events).

Entities are the individual elements of the system that must be defined. They are divided according to their movement in the system, if this entity flows within the system, it is called a dynamic entity, on the other hand if this is fixed or acts that are determined, it is called static entity. Entities are usually characterized with attributes that provide a value of a data assigned. These can be grouped into classes that are sets of entities of the same type and grouped according to their attributes.

Resources are elements of the system that provide a service to the entities. An entity can request one or more resource units, if this is not available, the entity have to start, for example in a pending queue or take another action. If the resource is available, it is used by the entity, held for using it on the time when it is necessity then made them available again. For instance, resources can be given by the worker who oversees the operation of the machine that can not work without worker. It is worth saying that an element of the model can also be considered as an entity or a resource generally, but this will depend on how the model is built.

An activity is an operation where the duration is known as prior at the beginning of the execution of the same activity. Duration can be a constant, a random value is generated related with probability of the distribution, given as an input or calculated on the basis of the other events that occur in the system. Delay is a period of time of indefinite duration time that is determined by the conditions of the system such as container in the queue.

First of all, empirical and mechanistic models must be classified: even if it is possible to say that all the models at a certain level are empirical, it often changes the level at which they are.

The empirical models are direct descriptions of the observed data and are constructed mainly for the purpose of describing the behavior of a system and to demonstrate the existence of relationships between selected variables without attempting to explain the nature of these relationships, in fact the structure and the level of detail is simple in general.

Mechanistic models try to explain the relationships between the elements of the modeled system: they try to describe the system at a certain number of levels below the one to which the estimation is made. In mechanistic models the representation of the lower levels of organization is often empirical, for example because the mechanisms that govern the processes at the lower level or for a deliberate choice are not clear.

The observed characteristic if the application has the time as a variable or not: in the first case dynamic model can be used, in the second static model.

In the first described type, the variable time will be explicitly present in the equations (i.e. differential) that identify the system or it is contemplated in the simulated process. In the second model, instead, time is not a variable. In fact these are not widely used, but in the dynamic models, there are entities that have a static behavior.

A further distinction is that linked to the state variables (as already mentioned above), that is between the continuous model, with the variables that vary continuously, and the discrete model, in which the value of the variables varies in well-defined moments of time (called states) [80].

Also there are other types of variables in the dynamic systems that differs from state variables. There are also rate variables that are auxiliary. First one is associated with the state variables to determine their rate of change as a function of time. The second is quantity that changes with time whose definition does not necessary to represent the behavior of the system but can be useful. The last one is called conditioning functions and they represent the outside factors of the system that works on the boundaries [25].

The last interesting distinction is between deterministic models and stochastic models: the first performs predictions provide numerical outputs without giving a measure of the probabilistic distribution of the results, the second, instead, contains procedures within them that take into account of probability distributions, together with elements that generate randomness for some states or variables To do this, models are often used to generate random numbers (i.e. *rand()* function in the C language) or estimates for the variability of the parameters that are used, verify the effect of this variability in the behavior of the system.

The Monte Carlo method is used for this types of model which consists to generate random numbers according to probability of the distributions to simulate the variability.

A simulator is chosen related with the specifications of the research. The simulation must be designed too. Before the implementation of the simulation, it is necessary to decide how to conduct it. Because the simulation is a process that evolves during its' realizations where the initial results help to conduct the simulation towards more complex configurations. There are also statistical problems. The first one is the determination of the length of the transient of the system before arriving to the stationary conditions, after that the data can be collected related to the system when it is operational. Another is the determination of the length of the simulation (duration) after the system has reached the equilibrium (in fact it must be always kept in mind that the simulation does not produce exact values of the measurement of the performance of the system as every single simulation can be seen as a statistical experiment that generates observations) These observations will be used to produce estimations of performance measurement and of course the precision will increase if the duration of the simulation increases [80].

The simulations will be performed and the results will be analyzed. A fundamental point is that each measure is accompanied by the "confidence interval" [80] (it can vary). These results can immediately show a better system configuration than the others.

These steps are standard for each type of simulator, i.e. for any of those described in the previous chapter.

In the particular case of the simulators and the discrete events, the procedure is the same but the transfer related with formulation of the model have to be paid more attention because changes can occur.

Before starting of the discussion on the formulation of the model, Simulators and discrete events have to be introduced. These serve to model of the dynamic systems where the state variables always assume discrete values and the transitions from one state to another take place in discrete moments corresponding to the occurrence of events [39]. This is different respect to what happens for the more conventional models of the dynamic systems in continuous state (continuous time or discrete time) where the state evolves over the time continuously with the differential equations or finite difference equations. Therefore, it is worth saying that the time in the simulation model does not progress continuously, but suddenly, from one moment to the next one, it is called a concept of ‘virtual’ time that is the one between two successive events. The last one is called delay [56].

Moreover, it is necessary to define a mechanism of the time progress to proceed simulated time from one value to another. The variable that in a simulation model provides the current value of the simulated time is called "simulation clock" [80], and there are two ways to define its progress: through turns the time to the next event (the common one is where the simulation clock is initialized to zero and is progressed at the time of the occurrence of the first of future events, then the system is going to update that takes into account the event that has happened, the times of the future event are updated and the procedure is iterated, inactive periods are not considered) or prefixed increase of the time progress (in this case the time is more similar with the real one, and it is worth saying that all the inactive periods of the systems are counted but it is less useful).

In the conclusion of the explanation of the systems of discrete events, single queue can be a good example [39].

Moreover, it is worth assuming that some points, such as knowledge of the arrival and departure process (all variables are random with the determined distribution), all clients in the system belong to the same class, the cycle is operative, the service policy is FIFO (First In – First Out), the cycle of the system can not be inactive if there are clients that wait for service.

Furthermore, the status of the system is defined with the variable $n'(t) = n(t) + \delta(t)$ that represents the total number of the client in the queue of the system($n(t)$ represents the sum of the number of the clients that wait in the queue and $\delta(t)$ clients in service)

It is necessary to identify the state variables of the system and the classes of the events that give a position to each transitions of the state for performing the simulation of the discrete events.

In the case that is applied on the thesis, the classes of the events are; e_1 = 'arrival of external clients', e_2 = 'end of service'. So the list of the classes of events are LCE = {e_1, e_2}. The list of the active event at time t is the dynamic list that is created for the classes of events where events are scheduled at time t (in other words, the occurrence is known at the give time), $LCA(t) = \{e_1, e_2\}$. Another dynamic list is the list of the times of the active events that indicates the time intervals between the times of the events that has already scheduled (presents in LCA) and instant time t, $LTEA(t) = \{T_1(t), T_2(t)\}$.

One of the most important thing for the simulations of the discrete events, all state variables between an event and the next one remain constant and the vector of the state variables are written as $\underline{x}(t^h)$, that contains all state variables in the interval time [t^h , t^{h+1}], the h-th event is obtained that occurred at t^h.

It is possible to say that the evolution of the state of the system takes place on the base of the transaction functions, $\underline{x}(t^h) = f\,[\underline{x}(t^{h-1}), e^h]$.

It is worth saying that pseudo-codes (Pseudocode is a detailed yet readable description of what a computer program or algorithm must do, expressed in a formally-styled natural language rather than in a programming language. Pseudocode is sometimes used as a detailed step in the process of developing a program) [55] can be written to describe function of events that belongs to LEA, whether the events are external or internal (whether they are dependent or not on the status of the system), 'end simulation have to be added to the event at the end.

Obviously it is necessary to realize the random variable with the best possible way.

Here is the sum of the development processes of the coherent model for the discrete event simulation [92].

- definition of the state variables;
- identification of the values that can be assumed by the state variables;
- realization of a simulated time measurement, "simulation clock", which records to elapse of the simulated time;
- implementation of a method to generate events randomly;
- identification of the state transitions generated by the events.

As it can be seen that there is a little difference with the model construction of the general case. There is no random variables in the first case.

1.2.2. Flowchart of the model

The flowchart is the best method to visualize immediately the behavior of the simulation model especially for the discrete events.

It allows to create graphical model (it has a graphical modelling language) of the operations (processes) that system performs, and represents the processes with the conventional shapes (the shapes depend on the program used) each shapes has logical meaning precisely and textual part describes the activity to be performed and the connections between the processes are connected with the arrows.

It is possible to create a model based on its characteristics, for this reason there are different types depend on what user wants to represent one thing rather than another. For example, the queue of the described below in Figure 1.

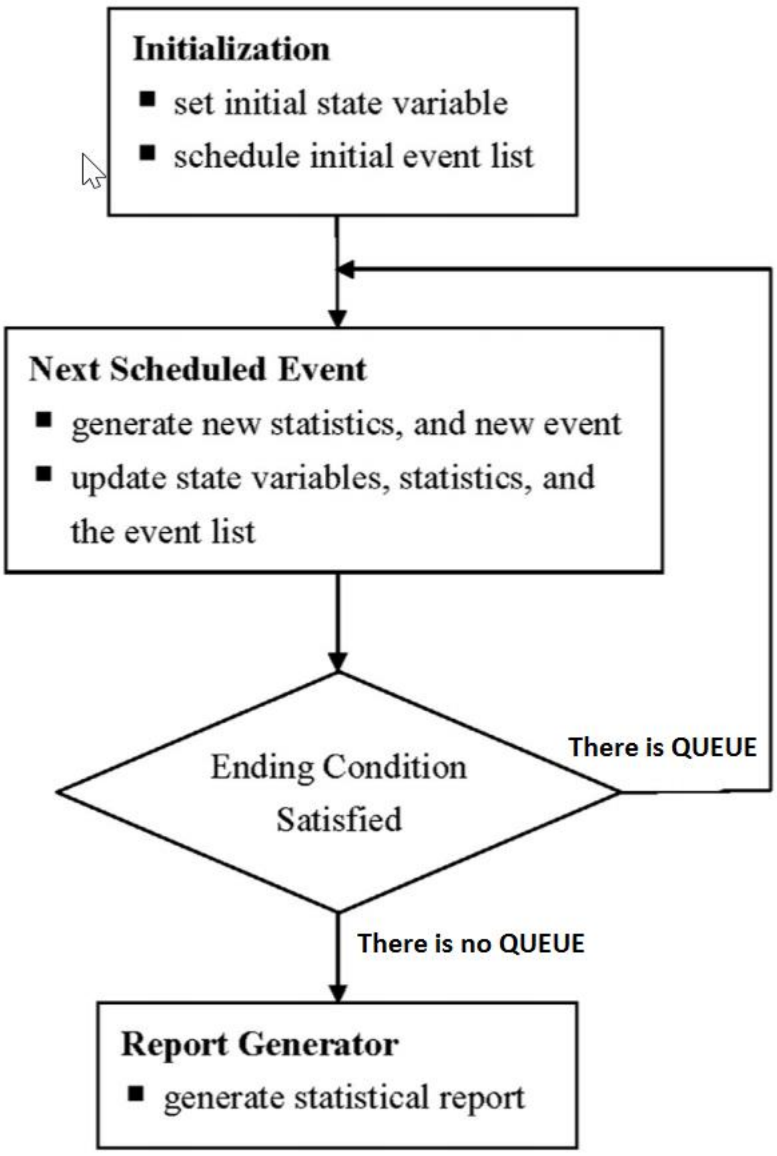

Initialization
set initial state variable
schedule initial event list
Next Scheduled Event
generate new statistics, and new event
update state variables, statistics, and the event list
Ending Condition Satisfied
There is QUEUE
There is no QUEUE
Report Generator
generate statistical report

Figure 1 Flowchart of the queue part of the model

It is created as seen in the figure, the events are indicated with the rectangles and the conditions with the diamond shape. If the condition is ok, the event can continue to the process.

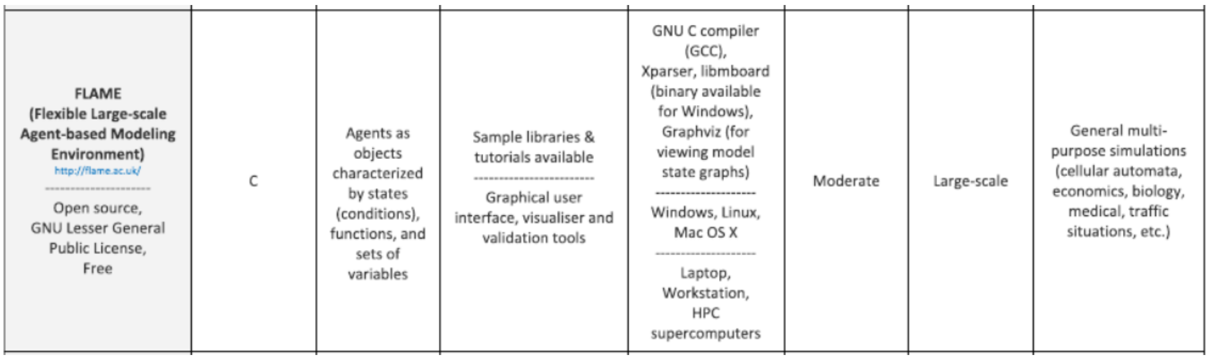
The flowcharts are very useful, extremely flexible and manageable, it is useable to describe the flow of the conditioned events for describing detailed algorithm, indicating all individual operations that is done because it helps to understand the model easily and another point that it does not require the knowledge of the programming language.

These are the reasons why all existing models use flowchart or using it for describing the functions of the system.

1.3. FLAME - Flexible Large-scale Agent Modelling Environment

In terms of model development effort against computational modelling strength/models' scalability level, FLAME has a good ratio, indeed it allows a high computational strength and a large-scale scalability level in the face of a moderate development effort.

The FLAME framework allows to develop agent-based models that can work on Large-scale parallel/Distributed computing clusters and high performance computers (HPCs). The computational model used to develop the agent-based models is called (extended finite) state machines. The FLAME tool permits to create an agent-based model defining software behavior by means of state machines with transition functions between those states. The agent/machine goes from a start state to an end state in a time step or iteration, passing through a several states with the own transition functions.



FLAME (Flexible Large-scale Agent-based Modeling Environment) http://flame.ac.uk/ ---------------------- Open source, GNU Lesser General Public License, Free	C	Agents as objects characterized by states (conditions), functions, and sets of variables	Sample libraries & tutorials available ------------------------ Graphical user interface, visualiser and validation tools	GNU C compiler (GCC), Xparser, libmboard (binary available for Windows), Graphviz (for viewing model state graphs) -------------------- Windows, Linux, Mac OS X -------------------- Laptop, Workstation, HPC supercomputers	Moderate	Large-scale	General multi-purpose simulations (cellular automata, economics, biology, medical, traffic situations, etc.)

Illustration 1 General view of FLAME

Each agent has variables into its memory. Transition functions allow to read and write to variables in the agent's memory and also read incoming messages and write outgoing messages. The agents communicate via messages.

The communication between agents drives the coordination of agent function execution since FLAME works with agent models in parallel and on different processors.

The Figure 2 shows an agent machine. It starts with a start state and finalizes with an end state and one transition function which takes it from one to the other state. The transition function can access to the agent memory (i.e. modifies the variables) and receives input messages and generates output messages.

This is the brief figure that explains an agent-based model that includes the individual steps for the creation of model. The individual steps are;

- Determine the agents and functions
- Determine the states that forward some order of function execution
- Determine the memory. It means the set of variables that access to the functions directly with possible conditions on these variables for the functions to be formed.
- Determine the messages as input and output for each function with the filters if they need.

When a model is created with agent functions with these criteria, C programming language is written as the source code of the implementation process. In the simulation phase, FLAME is used for the model description to create a program that execute agent and communication in parallel.

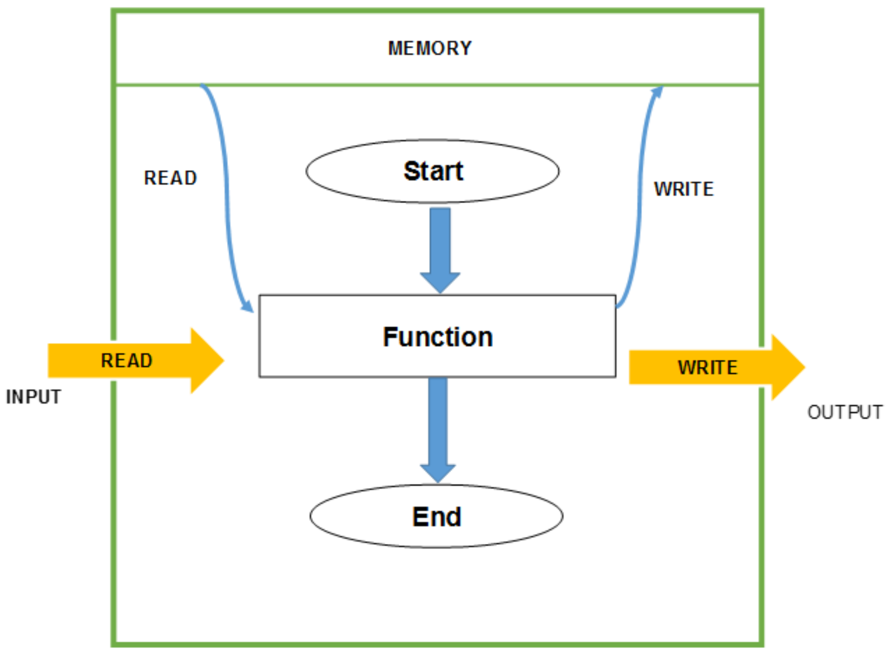


Figure 2 Agent Machine

The Figure 3 below shows the state mechanism of operation. The circles are the states and the rectangles are the functions. The states, representing the memory, sends continuously memory to the functions which are written on C files and represent the behavior of the agents for every process. If there are conditions, states decide OK or NOT. Functions perform or not, related with the answer of the conditions. This process continues until the end state. During the processes, agents communicate each other by means of the messages. Messages contain variables which are defined by XML files. XML files are the models or the sub-models.

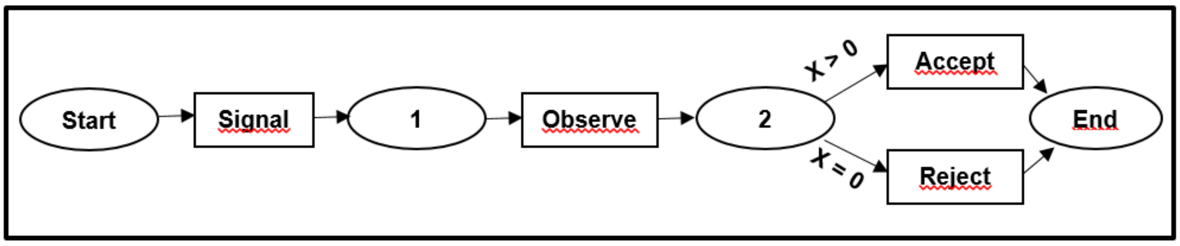


Figure 3 State Mechanism of Operation

There are a several steps for developing a model. They include: the identification of the agents and their functions; the identification of the states in order to establish and sequence of function execution; the identification of the variables into the agents' memory changeable by functions; the identification of the input messages and output messages of each function.

To produce the agent model and to be able to implement the simulations, the FLAME is chosen as a framework. It is a generic system of the agent model that can be used for the development of the applications in various research areas. The main idea is that it gives a capability to produce complete agent application that can be compiled and generated on the computing systems.

The model are created on the basis of the computational model that is called Finite State Machines (FSM). In this way with the FLAME framework can automatically generate simulation programs that execute aforementioned models with the efficient way.

It is worth mentioning that nearly all the explanations are taken from the documents cited on the biography in point [35].

The philosophy of FLAME is to specify the agent model as software behavior can be specified, as the model execution will be in the software. The behavior model is based on state machines, which are composed of a certain number of states (different configurations of the variables) with the transition functions that interconnect these states. There is an initial state for each agent. It will start when the execution starts, start state will be single and there will be only one final state too. The machine executes the functions by traversing status with using the transition functions until it reaches to end state. This process will happen to each agent/machine as one time step or iteration is completed.

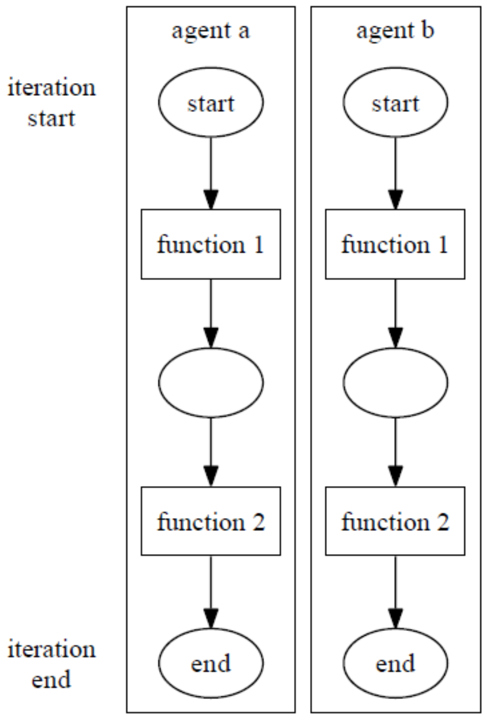


Figure 4 Agents with two functions

Figure 4 describes a model that has two agents and two functions for each agent. The functions run one after the other. A time step or iteration of the model is when each agent goes from their state to an end state. In addition, it is worth saying that there is no connection between these two agents.

Each agent has a memory that contains the variables that are modified, read or written by the transition functions. Agents can communicate with each other through messages, these can be incoming and then these messages will also be read or outgoing in the functions, and then the code to write the message will be presented in the function. Because FLAME can execute the agent model in parallel and on different processors the coordination and execution of the agent function depends precisely on the communication between agents.

The communication is synchronous with the agent model, which means that it happens during the execution of the model. For agents this means that if an agent function receives a certain type of messages, it can not be performed until all the messages of that type have been send and all ready to be read which means that no agent has priority for reading any input and all agents have access to the same message at the same time. Ultimately, it means that the order that agents are executed does no matter. Because agents can run anywhere on a super computer cluster FLAME uses a broadcast communication method. This means that agent can not send messages directly to another agent. Instead the receiving agent must filter its messages that is read.

The Genoa Port Model descriptions are formatted in XML (Extensible Markup Language) tag structures. FLAME allows to develop a model using multiple files included in one file.

Models can include:

- Different sub-models that can be exchanged and can be enabled or disabled inside the model.
- Environment which consist of constant variables, location of function files, time units, data types.

The constant variables are help to set up several simulations. The environment includes also the source code for the implementation of the agent functions and the time units which activate the agent functions. Finally, within the environment the data types are user defined data types used or not in a model.

Agent types which are defined by means of name, description, memory, functions.

The agent memory consist of variables defined by means of their type, C or user defined data types from the environment, a name, and a description. An agent function has a name, a description, the current state the agent has to be in, the next state the agent will transition to, a possible condition of the function transition, the possible input and output messages.

The current state and next state tags hold the names of states. This is the only place where states are defined. A function can have a condition on its transition. This condition can include conditions on the agent memory and also on any time units defined in the environment. It’s worth to note that for every agent it must be possible to become from the start state to an end state. Each possible transition must be mutually exclusive, and the order that the function conditions are tested is not defined. A function named ‘idle’ is available to be used for functions that do not require an implementation.

Message types which are defined by means of name, description, variables. Each message in a model has a type defined by a name and the variables included in the message. The message represent the communication among the agents. The communication via Message Boards and messages can be seen in the State Graph of the Model.

FLAME includes a parser program that parser a model described as XMML into simulation program source code (Figure 5). Then, this source code and the agent functions implementation source code are compiled. The compilation takes place using the compilation script 'Makefile' using the 'make' build automation tool. The program 'make' refers to the 'gcc' C compiler.

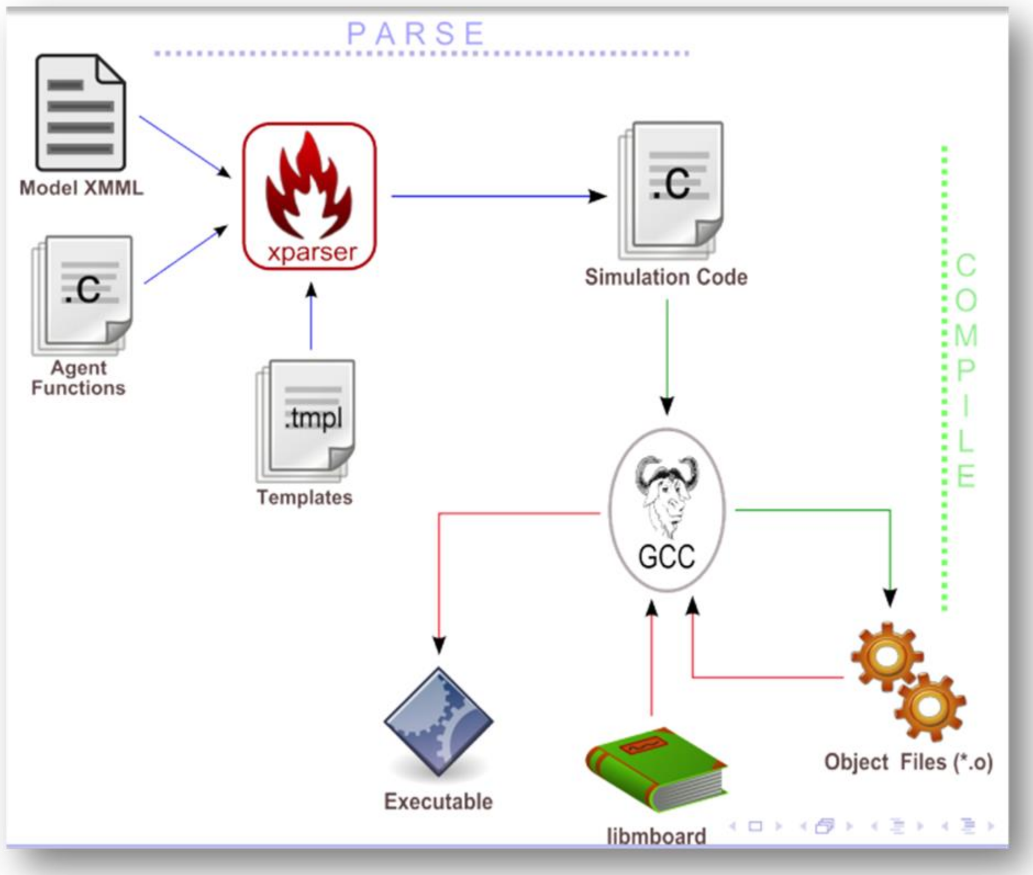


Figure 5 Schema of the simulation procedure

To compile a model the message board library is required. The compiled program is called 'main'. The parameters necessary to run a simulation are: the number of iterations to run for and the memory of agents which is a formatted XML file.

1.4. Model Construction

The model structures are written with XML structures (eXtensible Markup Language) to allow easy programming and readability, as well as to improve collaboration between developers who interact in the drafting of the model. The model has a well-defined XML structure available on the site [35].

XML is a metalanguage for the definition of markup languages that is a marker language (it is a set of rules that describe the representation mechanisms of a text that use standardized conventions, can be used on supports) based on a syntactic mechanism which allows to define and control significant elements contained in the document or in the text. The name indicates that it is an extensible markup language (extensible) as it allows to create personalized tag [85].

The start and the end of a template file must be formatted as follows:

```
<!DOCTYPE xmodel SYSTEM "">
<xmodel version="2" >
 <name>Port</name>
 <version>Version 0.1, 19/10/2018</version>
 <description>Authors: Onur Senturk</description>
 …
</xmodel version="2" >
```

where name is the model name, version is the version and description is the description of the model.

The model may also contain:

- Other models which can be enabled or disabled.
- The environment that contains constant values, location of the function files, time units, data types.
- Agent types that include name, description, memory, functions.
- Message types that include name, description and variables.

1.4.1. Sub-models

The agent models can be defined in multiple files that FLAME reads a model from multiple files as if the model were defined in a file. This feature allows you to easily enable or disable different parts of the model for example agents can be enabled or disabled separately. In addition, if the model has a different versions of a sub-models, these are can be exchanged, or the sub-model can be disabled to examine how it works without that part, how it changes the model. The following tags indicate the inclusion of two models, one enabled and one disabled (the location of the sub-models is connected with the location of the model):

```
<models>
 <model>
  <file>AgentMaritime/model.xml</file>
  <enabled>true</enabled>
 </model>
 <model>
    <file>Customs/model.xml</file>
    <enabled>true</enabled>
  </model>
<model>
    <file>CustomClearanceAgent/model.xml</file>
    <enabled>true</enabled>
  </model>
<model>
    <file>Terminal/model.xml</file>
    <enabled>true</enabled>
  </model>
    <model>
    <file>DeliveryService/model.xml</file>
    <enabled>true</enabled>
  </model>
 <model>
    <file>Deliverer/model.xml</file>
    <enabled>true</enabled>
  </model>
   <model>
    <file>Gate/model.xml</file>
    <enabled>true</enabled>
  </model>
  <model>
    <file>Financepolice/model.xml</file>
    <enabled>true</enabled>
  </model>
  <model>
    <file>Area/model.xml</file>
    <enabled>true</enabled>
  </model>
</models>
```

Figure 6 Sub-Models (Agents' models)

In the port model, there are nine sub-models. It is worth saying that the location of any sub model should be relative to the original model xmml file. For example, main port model that includes nine sub-models that are in the same xmml.

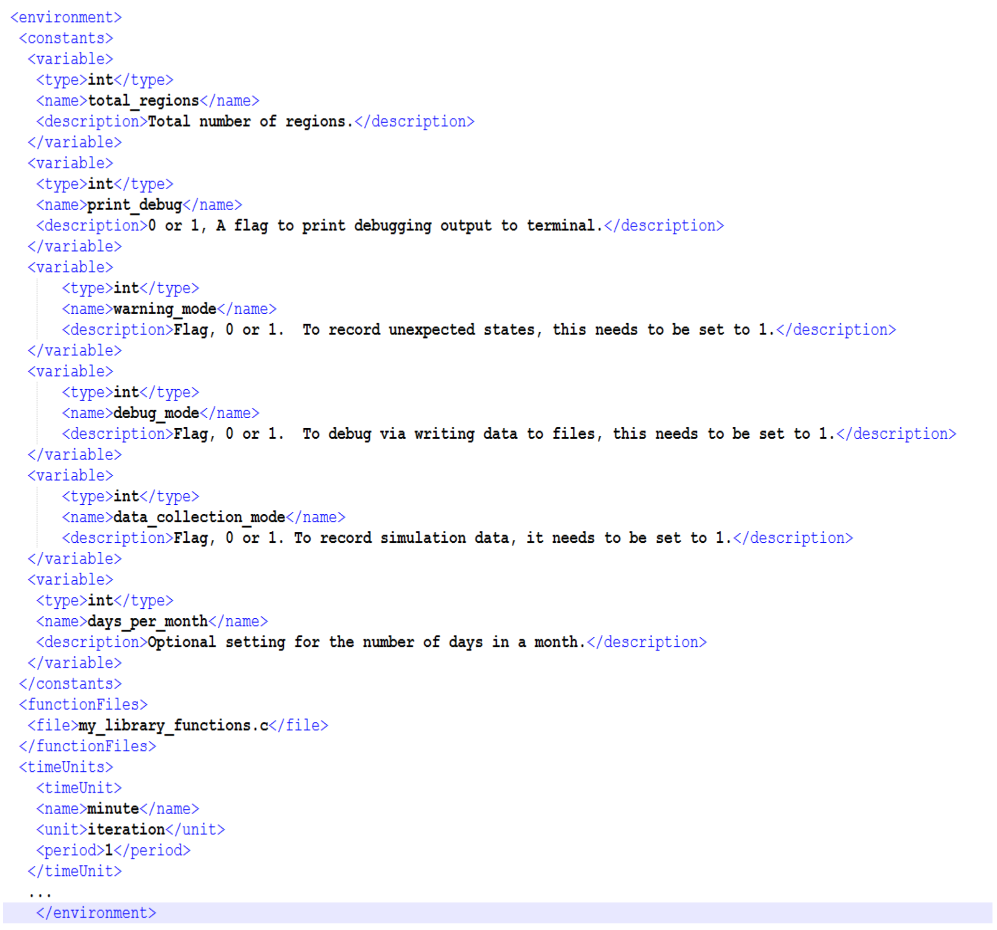
1.4.2. Model Environment

The model environment contains the information required by the model that is not part of an agent or message. They include:

- *environmental constants(variables)*, used to easily set up different simulations;
- *location of the function files*, that is the paths of these files such as agent functions;
- *time units*, used to be able to use the functions of the agent dependent on time periods;
- *data types*, which are user-defined data types that are used by agent memory or message variables.

This notion of environment is not that of an environment that is part of a model in which agents would interact because, since everything that can change in a model must be represented by an agent, if a model includes a modifiable environment with the which agents can interact this in itself must be represented by an agent.

Figure 7 shows us environment includes variables, functions, time units etc.

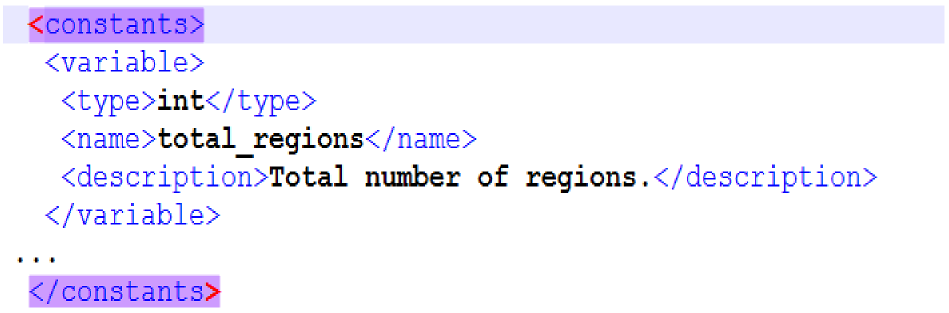


```
<environment>
 <constants>
  <variable>
   <type>int</type>
   <name>total_regions</name>
   <description>Total number of regions.</description>
  </variable>
  <variable>
   <type>int</type>
   <name>print_debug</name>
   <description>0 or 1, A flag to print debugging output to terminal.</description>
  </variable>
  <variable>
      <type>int</type>
      <name>warning_mode</name>
      <description>Flag, 0 or 1.  To record unexpected states, this needs to be set to 1.</description>
  </variable>
  <variable>
      <type>int</type>
      <name>debug_mode</name>
      <description>Flag, 0 or 1.  To debug via writing data to files, this needs to be set to 1.</description>
  </variable>
  <variable>
      <type>int</type>
      <name>data_collection_mode</name>
      <description>Flag, 0 or 1. To record simulation data, it needs to be set to 1.</description>
  </variable>
  <variable>
   <type>int</type>
   <name>days_per_month</name>
   <description>Optional setting for the number of days in a month.</description>
  </variable>
 </constants>
 <functionFiles>
  <file>my_library_functions.c</file>
 </functionFiles>
 <timeUnits>
   <timeUnit>
   <name>minute</name>
   <unit>iteration</unit>
   <period>1</period>
  </timeUnit>
  ...
   </environment>
```

Figure 7 Environment

1.4.3. Constant Variables

The *environmental constants* (*Constant Variables*) that must be set at the beginning of the simulation and are not modified. They are used to set different settings, such as debugging functions or similar. Their structure is as follows:

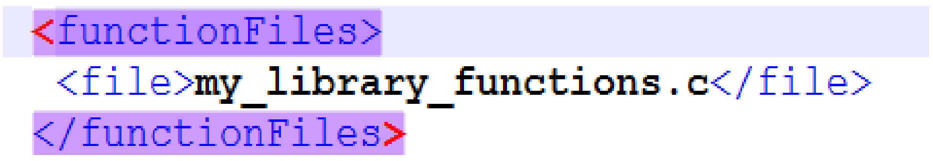


```
<constants>
  <variable>
    <type>int</type>
    <name>total_regions</name>
    <description>Total number of regions.</description>
  </variable>
...
</constants>
```

Figure 8 Constants

1.4.4. Function Files

The locations of function files are paths to files that contain agent functions. They can be located in the compilation script (Makefile) of the model. These are related to the location of the model and are structured as follows:



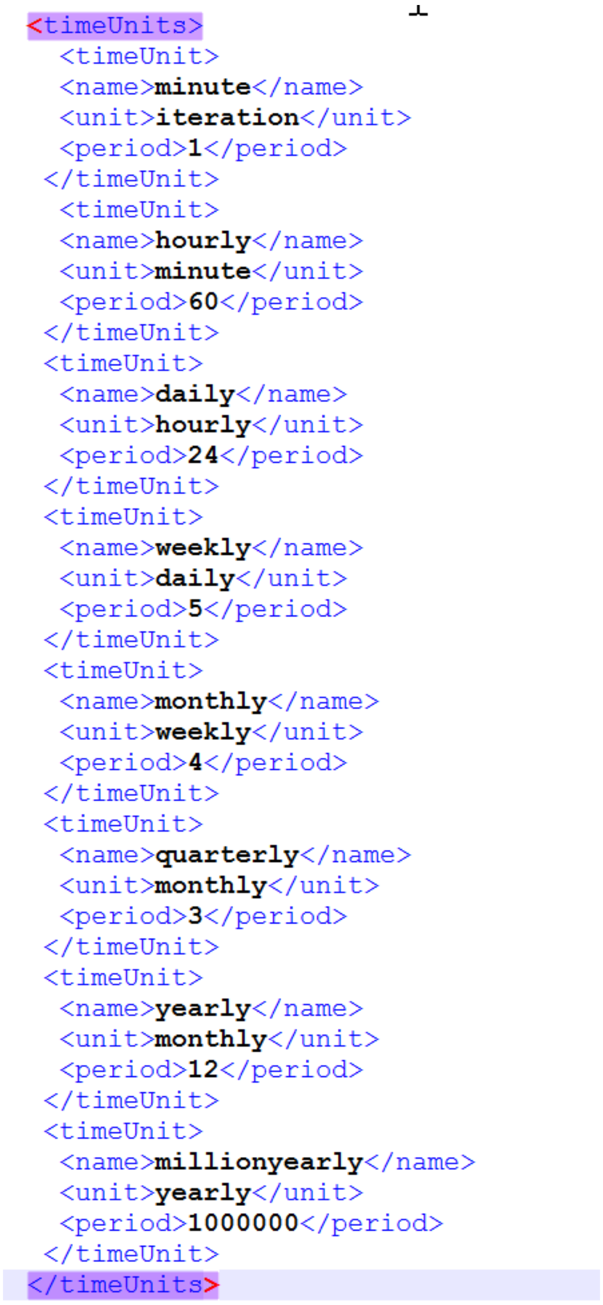
```
<functionFiles>
  <file>my_library_functions.c</file>
</functionFiles>
```

Figure 9 Functions

1.4.5. Time Units

Time units are used to define the periods of time in which the functions of the agents act within. For example, a model that uses a calendar-based time system might have the day as a minor step of time, that is an iteration. You can then use these to define other time units, such as weeks, months and years. They consist of name of the units, Unit contains iteration or other defined time units, period the length of the time unit.

As it is seen in Figure 10 Time Units structure, one minute equals one iteration.



```
<timeUnits>
    <timeUnit>
    <name>minute</name>
    <unit>iteration</unit>
    <period>1</period>
  </timeUnit>
    <timeUnit>
    <name>hourly</name>
    <unit>minute</unit>
    <period>60</period>
  </timeUnit>
  <timeUnit>
    <name>daily</name>
    <unit>hourly</unit>
    <period>24</period>
  </timeUnit>
  <timeUnit>
    <name>weekly</name>
    <unit>daily</unit>
    <period>5</period>
  </timeUnit>
  <timeUnit>
    <name>monthly</name>
    <unit>weekly</unit>
    <period>4</period>
  </timeUnit>
  <timeUnit>
    <name>quarterly</name>
    <unit>monthly</unit>
    <period>3</period>
  </timeUnit>
  <timeUnit>
    <name>yearly</name>
    <unit>monthly</unit>
    <period>12</period>
  </timeUnit>
  <timeUnit>
    <name>millionyearly</name>
    <unit>yearly</unit>
    <period>1000000</period>
  </timeUnit>
</timeUnits>
```

Figure 10 Time Units

1.4.6. Data Types

Data types are new types of user-defined data. The individual variables can be basic C types (such as int, float or double), static arrays (specifying their size, as variable_name [size]) and dynamic arrays (adding the suffix "_array" to the data type), and define so:

```
<variables>
  <variable>
    <type>access_type</type>
    <name>access</name>
    <description></description>
  </variable>

  <variable>
    <type>int</type>
  <name>transportationready</name>
    <description></description>
  </variable>

  <variable>
    <type>gateout_type</type>
    <name>gateout</name>
    <description></description>
  </variable>
</variables>
```

Figure 11 Data Types

Data types can also contain other created types.

1.4.7. Agents

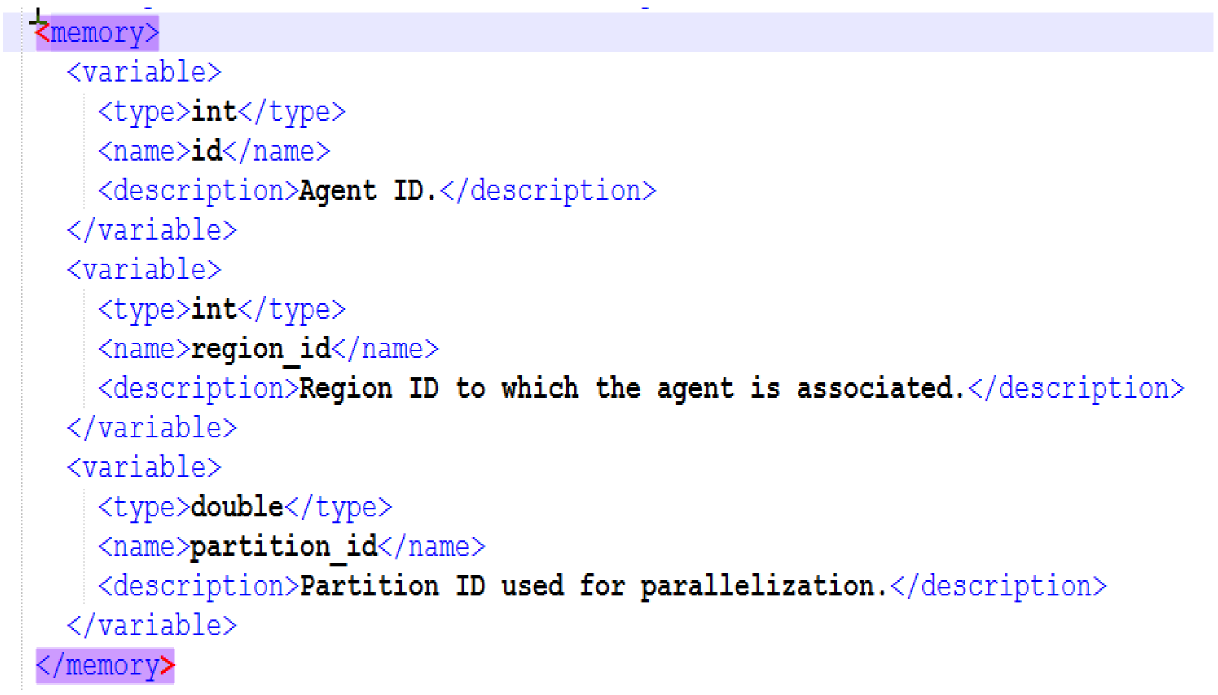
Agents are defined by name, description, memory and functions.

```
<agents>
   <xagent>
        <name>CustomClearanceAgent</name>
        <description>rappresenta l'importatore (=cliente)</description>
        <memory>
            <variable>
                <type>int</type>
                <name>id</name>
                <description>Agent ID.</description>
            </variable>
            <variable>
                <type>int</type>
                <name>region_id</name>
                <description>Region ID to which the agent is associated.</description>
            </variable>
            <variable>
                <type>int</type>
                <name>partition_id</name>
                <description>Partition ID used for parallelization.</description>
            </variable>
        </memory>
        <functions/>
        ...
    </xagent>
</agents>
```

Figure 12 Agents (Sample of CustomClearanceAgent)

1.4.8. Agent Memory

The *memory* defines the agent variables, defined by the data type, which can be one of the C data types or one of the data types that are defined by the user in the environment, a name and a description:

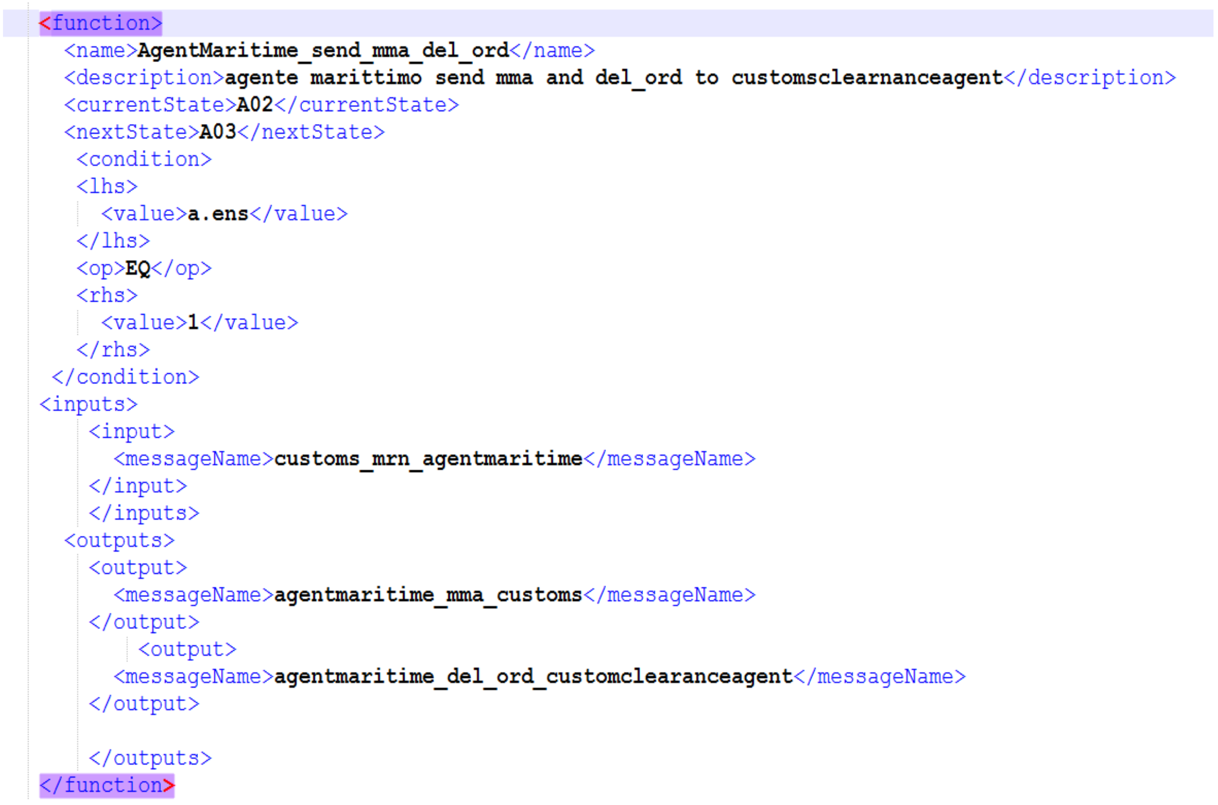


```
<memory>
  <variable>
    <type>int</type>
    <name>id</name>
    <description>Agent ID.</description>
  </variable>
  <variable>
    <type>int</type>
    <name>region_id</name>
    <description>Region ID to which the agent is associated.</description>
  </variable>
  <variable>
    <type>double</type>
    <name>partition_id</name>
    <description>Partition ID used for parallelization.</description>
  </variable>
</memory>
```

Figure 13 Agent Memory

1.4.9. Agent Functions

The *agent functions* contain the name of the function, which must correspond to a name of an implemented function and must be univocal on the whole model. A description; the current state, the agent has to be in. i.e. the one in which the agent must be to start the function; the next state that is in which the agent will pass; any conditions that may affect the transition of the function or another; the possible incoming messages is called inputs and the possible output messages is called outputs. In Figure 14 show us the structure of the Agent function.



```
<function>
  <name>AgentMaritime_send_mma_del_ord</name>
  <description>agente marittimo send mma and del_ord to customsclearnanceagent</description>
  <currentState>A02</currentState>
  <nextState>A03</nextState>
   <condition>
   <lhs>
     <value>a.ens</value>
   </lhs>
   <op>EQ</op>
   <rhs>
     <value>1</value>
   </rhs>
 </condition>
<inputs>
    <input>
      <messageName>customs_mrn_agentmaritime</messageName>
    </input>
    </inputs>
  <outputs>
    <output>
      <messageName>agentmaritime_mma_customs</messageName>
    </output>
        <output>
      <messageName>agentmaritime_del_ord_customclearanceagent</messageName>
    </output>

    </outputs>
</function>
```

Figure 14 Agent Functions

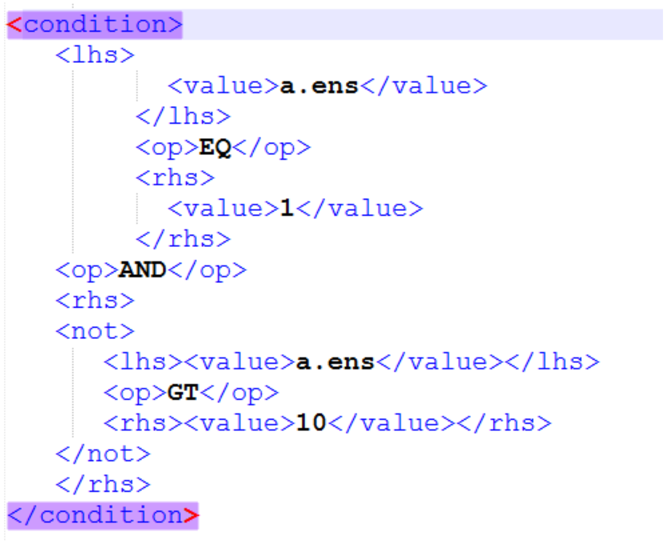
State names must be placed in the current state and next state, and it is the only place in which they are defined. The names of the states must coordinate with the other functions and the other states in order to create the transitional graph, which starts from the start state and arrives at the end state (there may be more than one end state).

Agent functions can also have conditions on the transitions. The conditions include conditions on the agent memory and on any time units that is defined in the environment. Outgoing transition with any condition at any state must be possible also for a transition to happen. It means that the condition must be possible for each agent from start state to end state. The next state will be different. Each condition must be mutually exclusive. The order of these is not defined and can be tested without any problems in any order. The 'Idle' function is available for the functions that does not need to implement.

Conditions have the form (with tags) <lhs> </ lhs>, <op> </ op>, <rhs> </ rhs>, where lhs is the variable or value of the left side, op is the operator of comparison and rhs is the variable or value of the right side. The variables can also be those of the agents: in this case they must be preceded by the letter 'a'.

The comparison and logical operators are: EQ (equal to), NEQ (not equal to), LEQ (less than or equal to), GEQ (greater than or equal to), LT (less than), GT (greater than), IN (the integer on the left is a member of the array of int on the right), logical AND, logical OR.

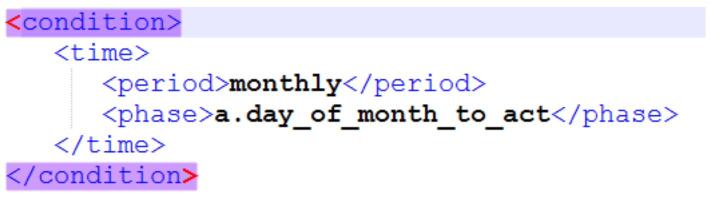
There is still the NOT operator that can be added to one of the above to deny it, and it is possible to write conditions within the individual parts lhs or rhs, as in this example:



```
<condition>
   <lhs>
          <value>a.ens</value>
        </lhs>
        <op>EQ</op>
        <rhs>
          <value>1</value>
        </rhs>
   <op>AND</op>
   <rhs>
   <not>
      <lhs><value>a.ens</value></lhs>
      <op>GT</op>
      <rhs><value>10</value></rhs>
   </not>
   </rhs>
</condition>
```

Figure 15 Comparison and logical operators

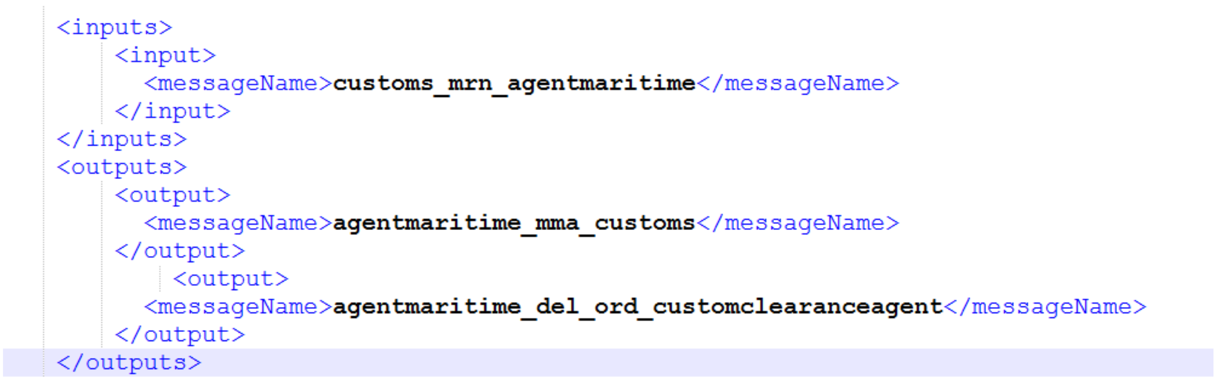
A condition can also depend on any time units which are described in the environment. For example, the following condition is true when the agent variable "day_of_month_to_act" is equal to the number of iterations that occurred from the beginning, i.e. 20 iterations defined in the time unit, i.e. the "monthly" period:



```
<condition>
   <time>
      <period>monthly</period>
      <phase>a.day_of_month_to_act</phase>
   </time>
</condition>
```

Figure 16 Monthly period-condition

The functions can have incoming as input and outgoing as output messages. This example describes a function that has message customs_mrn_agentmaritime as input, and message agentmaritime_mma_customs and agentmaritime_del_ord_customclearanceagent as an output in the Figure 17.

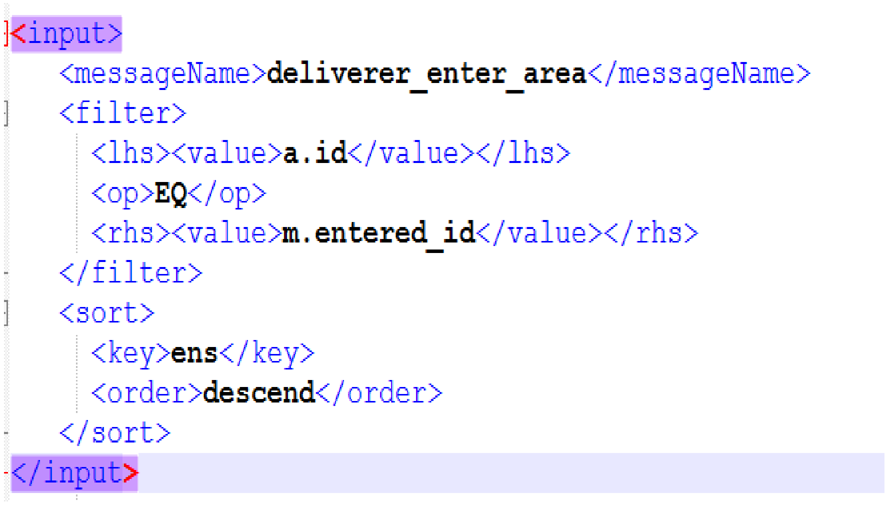


```
<inputs>
    <input>
      <messageName>customs_mrn_agentmaritime</messageName>
    </input>
</inputs>
<outputs>
    <output>
      <messageName>agentmaritime_mma_customs</messageName>
    </output>
        <output>
      <messageName>agentmaritime_del_ord_customclearanceagent</messageName>
    </output>
</outputs>
```

Figure 17 Input and output messages

Messages can be "filtered" through the conditions. Variables can be preceded by a 'm' if they are variables of the message. Another possibility is to sort incoming messages based on a variable that we call "key".

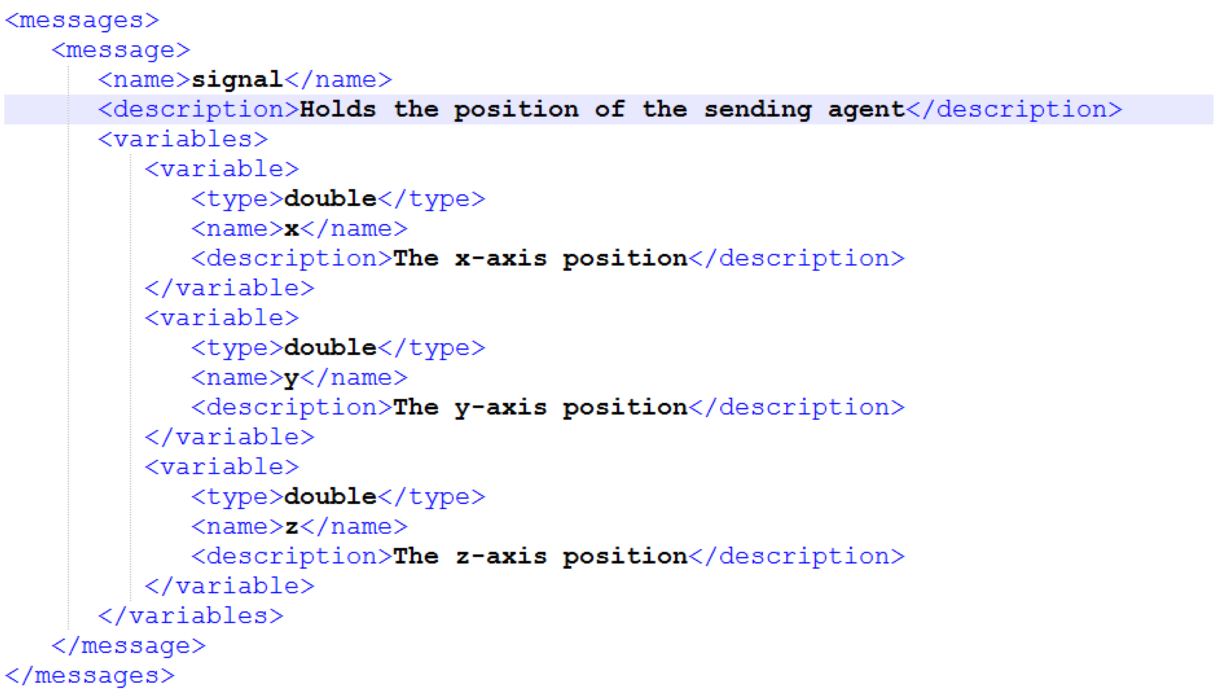
The following filter saves only messages where the agent variable "id" is equal to the "entered_id" message variable and sorts them in descending order based on their "ens" value. There is another option to order. It is ascend command.



```
<input>
    <messageName>deliverer_enter_area</messageName>
    <filter>
      <lhs><value>a.id</value></lhs>
      <op>EQ</op>
      <rhs><value>m.entered_id</value></rhs>
    </filter>
    <sort>
      <key>ens</key>
      <order>descend</order>
    </sort>
</input>
```

Figure 18 Sorting values of the messages

Messages are defined in the model by a name and the variables that the message includes (these can not be dynamic arrays). The following example shows a message called "signal" that passes the position of a point in 3D:



```xml
<messages>
   <message>
      <name>signal</name>
      <description>Holds the position of the sending agent</description>
      <variables>
         <variable>
            <type>double</type>
            <name>x</name>
            <description>The x-axis position</description>
         </variable>
         <variable>
            <type>double</type>
            <name>y</name>
            <description>The y-axis position</description>
         </variable>
         <variable>
            <type>double</type>
            <name>z</name>
            <description>The z-axis position</description>
         </variable>
      </variables>
   </message>
</messages>
```

Figure 19 Position of a point in 3D

1.5. Starting point of Implementation

The functions of each agent are written in C in separate files, ending with the suffix '.c'. Each file must include two header files, one for the general framework and one for the specific agent to which the functions refer. If these are for different agents, they can not be contained in the same file, so at the beginning of each file two headers are needed.

#include "../header.h"

#include "../AgentMaritime_agent_header.h"

So the functions will be written in C with the usual structure. They have no parameters.

You can add agents with the function code using the add_agentname_agent statement (var1, ... varN); which will add at the next iteration, a new agent of that type and with the values written there.

It is possible to access the agent variables on the functions that are indicated with the name of the variable in upper case (AGENT_VARIABLE), if this is a single value. It must be written the name of the array in uppercase and in square brackets the number of the element that is wanted to obtain. If it is a static array (MY_STATIC_ARRAY [index]). On the other hand, if it is a dynamic array, size of the function have to be calculated with using the '.size' function (MY_DYNAMIC_ARRAY.size) and then can be accessed the desired position MY_DYNAMIC_ARRAY.array [index]. It is also possible to access a variable within a model data type with the '.variablename' function (MY_DATA_TYPE.variablename).

Special instructions and syntax are used to send and receive messages. The following macros are used to receive to read them.

```
START_MESSAGENAME_MESSAGE_LOOP

messagename_message->variablename

FINISH_MESSAGENAME_MESSAGE_LOOP
```

Where the "MESSAGENAME" the name of the message will be inserted and the arrow operator (->) will be used to access the desired variables within the message.

To send them, use this syntax instead:

```
add_messagename_message (var1, .. varN);
```

Chapter 2

Development & Execution of the Model

2.1. Development Process

The model has been developed following the dictates given by the port authority and they are addressed, are made them fit with the parameters of the FLAME framework.

The agents, decided to develop, are the same as those described in the port process: AgentMaritime (marine agent), Area (terminal container storage area), CustomClearanceAgent, Customs, Deliverer (carrier), DeliveryService (forwarder), FinancePolice, Gate, and Terminal.

The memory variables, the functions of the model and the messages that will be explained in this section both for the environment and the individual agents.

First of all, it is worth taking that a new type of data named ‘mrn_type’ that is the Movement Reference Number, are created. It contains all the significant variables for the container. In particular it is an array that has 40 elements (all int) in the following order.

1. *id*: the number of the container assigned by Customs when it is processed.
2. *arrivalid*: the number of the container assigned by Customs when it arrives.
3. *arrivalid_time*: iteration / minute of arrival of the container in Customs.
4. *arrivalid_time_end*: iteration / minute in which the container in Customs begins to be processed.
5. *mma*: is a flag that if it is one it indicates that the MMA for the container is ready, if it is zero, not.
6. *mma_time* : number of iterations / minutes needed to produce MMA.
7. *mma_time_end* : iteration / minute in ‘when the MMA is ready’.
8. *a3 :* is a flag. If it is 1 it means that the A3 document for the container is ready.
9. *customdeclaration* : is a flag. if it is 1, it means that the Customs declaration for the container is ready.
10. *customdeclaration_time* : number of iterations / minutes needed to produce the Customs declaration.
11. *customdeclaration_time_end* : iteration / minute in ‘the Customs Statement is ready’.

12. *del_ord* : is a flag. If it is 1 it means that the delivery order for the container is ready.
13. *del_ord_time* : number of iterations / minutes needed to produce the Shipping Order.
14. *del_ord_time_end* : iteration / minute in 'the Shipping Order is ready'.
15. *clearcode* : is a flag that indicates whether or not you have the authorization code for the container. It is passed from Customs to CustomClearanceAgent.
16. *bookforpickup* : is a flag. If it is 1, it means that the delivery order (Buono di Consegna) for the container is ready.
17. *bookforpickup_time* : number of iterations / minutes needed to produce the delivery order.
18. *bookforpickup_time_end* : iteration / minute in 'the Delivery Order is ready'.
19. *pickup* : is a flag. If it is 1 it means that it is possible to withdraw the container. Terminal sends it to CustomClearanceAgent.
20. *pickup_time* : number of iterations / minutes necessary to authorize the withdrawal.
21. *pickup_time_end* : iteration / minute in 'it will be possible to withdraw the container'.
22. *ot* : is a flag. If it is 1 it indicates that the transport order for the container is ready.
23. *ot_time* : number of iterations / minutes needed to produce the transport order.
24. *ot_time_end* : iteration / minute in which the transport order is ready.
25. *ot_request*: is a flag. If it is 1 it indicates that the transport order is required for the container.
26. *permissionexit* : is a flag. If it is 1 indicates that the container is authorized to exit.
27. *w* : is a document required for DV.
28. *dv* : document passed from the Gate to the Deliverer.
29. *d* : is a flag. If it is 1 it means that documentation has been sent for the container on its date.
30. *no_request* : is a flag. It is Nulla Osta. If it is 1, it means that there is no need for a container.
31. *no* : it is a flag. If it is 1, it means that Nulla Osta document for the container is ready.
32. *no_time* : number of iterations / minutes necessary to produce the Nulla Osta.
33. *no_time_end* : iteration / minute in 'the Nulla Osta is ready'.
34. *gatein* : is a flag. If it is 1 it means that the container has entered the Terminal.
35. *gatein_time* : number of iterations / minutes necessary for the container to enter the Terminal.
36. *gatein_time_end* : iteration / minute in 'container enters the Terminal'.

37. *gateout* : is a flag. If it is 1 it means that the container can exit the Terminal and terminate the process.
38. *container* : is a flag. If it is 1 it means that the container has been picked up.
39. *loading_container* : is a flag: if it is 1 it means that the container is being loaded.
40. *finish_loading_container* : is a flag. If it is 1 it means that the loading of the container has ended.

The "mrn_type" type is presented in every agent and is used (as will be seen) for the variables: mrn, mrntemp, agentmaritimetctrn_array, vectormrn_list (last two are arrays of mrn_type).

2.1.1. Constant variables (Environmental constants)

All these variables must be entered manually in PopGUI. These variables are;

int total_regions : total number of regions. (int represents integer value)

int print_debug : it is a flag, 0 or 1. To print debug output in the terminal Agent.

int warning_mode : it is a flag, 0 or 1. To record unexpected situation.

int debug_mode : it is a flag, 0 or 1. To print .txt files during the execution of the program.

int data_collection_mode : it is a flag, 0 or 1. To register data of the simulations.

int days_per_month : This is an optional setting for the days of the month.

There are also temporal units. These represents time of the whole simulation.

- *minute*: 1 x iteration.
- *hourly*: 60 x minute.
- *daily*: 24 x hourly.
- *weekly*: 5 x daily.
- *monthly*: 4 x weekly.
- *quarterly*: 3 x monthly.
- *yearly*: 12 x monthly.
- *millionyearly*: 1000000 x yearly.

2.1.2. Global Variables of Agents

int id : It is the identification number of the agent.

int region_id : This is the identification number of the region.

int partition_id : It is the identification number of the parallelization (it is used in the case of parallel simulations on High Performance Computer).

MIN : It is the number of iterations, it is present in each table that will be generated in the database.

2.2. Agents' implementation

The Genoa Port Model is populated by the following agents' type: Agent Maritime, Customs, Custom Clearance Agent, Delivery Service, Deliverer, Terminal, Finance Police, Gate and Terminal Area. The Model implementation occurs using the Data Types, the Memory Variables, the Functions, the Messages reported in the following tables for each Agent included in the Port Model.

Tables from Table 1 to Table 9 summarize the memory variables, functions and messages for all the agent presented in the model.

2.2.1. Variables, Functions and Messages of Agent Maritime

AGENT MARITIME	***Name: Description***
Data Types	***mrn_type:*** includes variables that we used during the simulations
Memory Variables	***int ens:*** ens document which is send by Agent Maritime. It is called Entry Summary Declaration. (ENS document for the container - if it is 1 the container is there, if it is 0 there is not)
	int ensl: Similar like ens. It is used to send message.
	int ens_max: Maximum number of containers that the Maritime Agent can process at the same time (chosen by the user in PopGUI).
	int tipo_sdoganamento: Type of chosen customs clearance.
	mrn_type mrn: mrn document which is send by Agent Maritime. It is called Movement Reference Number. MRN doc for container.
	mrn_type mrntemp: MRN document for temporary containers.
	int sizeend: specified to define iterations. Number of containers in the agent.
	int_array ens_list: Array of the ENS is present in the agent.
	int_array agentmaritimevectin_list: Array of the ids of the containers entered into the agent. It means the list for the container arrived.
	int_array agentmaritimevectin2_list: Array of the ids of the containers entered into the agent.
	int_array agentmaritimevectout_list: Array of the container ids exited by the agent.List for the container got out.
	int_array agentmaritimevectout2_list: Array of the container ids exited by the agent

	mrn_type_array agentmaritimevectormrn_array: MRN array of containers within agent.
	mrn_type_array vectormrn_list: Similar to the one above.
Functions	***AgentMaritime_elaborate_ens:*** Agent Maritime prepares ENS for every container. It means agent creates ENS.
	AgentMaritime_send_ens: Agent Maritime send ens to Customs Agent.
	AgentMaritime_receives_mrn: Agent Maritime receives MRN from Customs Agent. The MRNs of the containers processed by Customs are received and is assigned mma_time and del_ord_time.
	AgentMaritime_send_mma_del_ord: Agent Maritime sends MMA (Manifesto Merci in Arrivo) and DO(Delivery Order) to Customs and CCA. First check that mma_time_end and del_ord_time_end are not equal to MIN: if yes, mma and del_ord are set to 1. Then the MRNs are sent to Customs and CCA.
Messages	***agentmaritime_ens_customs***: output message ens which is send by AgentMaritime to the Customs Agent with the ENS document. Message contains ENS.
	agentmaritime_sizeend_customs: output message sizeend which is send to Custom by AgentMaritime.
	agentmaritime_mma_customs: output message mma which is send by AgentMaritime to the Customs Agent. Message contains MMA.
	agentmaritime_del_ord_customclearanceagent: output message delivery order which is send by AgentMaritime to the CustomsClearanceAgent. Message contains Delivery Order.

Table 1 Agent Maritime Variables, Functions and Messages

2.2.2. Variables, Functions and Messages of Customs Agent

CUSTOMS	
Data Types	***customsvectormrn_array: Dynamic array for the MRN with the mrn_type variable.***
Memory Variables	***int customs_work_max:*** Maximum number of containers that the Customs can work at the same time (chosen by the user in PopGUI).
	int count: A counter used to assign id. It calculates to count container
	int arrivalcount: A counter that serves to assign arrivalid. To calculate containers arrived.
	mrn_type mrn: Container's MRN document.
	int ens: ENS Document for the container**.**
	int sizeend: Number of containers in the agent.
	mrn_type mrntemp: Temporary MRN document.
	mrn_type_array customsvectormrn_list: Array of the MRNs of the containers within the agent.
	mrn_type_array vectormrn_list: Similar to the one above.
	mrn_type mrntemp_customs: Temporary MRN document.
	int_array ensc_list: Array of the ENS of the containers that are in the queue.
	int_array ensc_arrivalid_list: Array of arrivalid of containers that are in the queue.
	int_array ensc_time_list: Array of arrivalid_time of the containers in the queue.
	int_array customsvectin_list: Array of the ids of the containers that entered into the agent.
	int_array customsvectin2_list: Array of the ids of the containers entered into the agent.
	int_array customsvectout_list: Array of the container ids exited by the agent.
	int_array customsvectout2_list: Array of the container ids exited by the agent.

Functions	***Customs_receives_ens:*** function that ens received. It is worth saying that the customs receive the containers.
	Customs_send_mrn: function that customs sends MRN to AgentMaritime.
	Customs_receives_mma: function that MMA received. The customs receives mma and also delivery order or shipping order.
	Customs_send_a3: function that A3 is sent to Custom Clearance Agent by Customs.
	Customs_elaborate_declaration: function for Custom declaration which is send by CCA.
	Customs_allow_clearance: function to allow clearance.The customs sends the Authorization Code to the customs clearance agent.
Messages	***customs_mrn_agentmaritime:*** message to send mrn.
	customs_a3_customclearanceagent: message to send a3.
	customs_clearcode_customclearanceagent: message to send clearcode.

Table 2 Customs' Variables, Functions and Messages

2.2.3. Variables, Functions and Messages of Customs Clearance Agent

CUSTOM CLEARANCE AGENT	
Data Types	***customclearanceagentvectormrn_array:*** *array for MRN in CCA.*
Memory Variables	***mrn_type_array customclearanceagentvectormrn_list:*** Array of container MRNs in the agent.
	mrn_type_array cca_list: Similar to the one above.
	int_array customclearanceagentvectout_list: Array of outbound container ids.
	mrn_type mrntemp_cca: Temporary MRN document.
	int answer: Specifies the type of response.
	int sizeend: Number of containers in the agent.
	int_array ccavectin_list: Array of the containers ids that are entered to the agent. To control whether the containers is in or not. It is mrn_type array.

	int_array ccavectin2_list: Array of the containers' ids that entered to the agent. Same as above.
	int_array ccavectout_list: Array of the containers' ids that is exited by the agent. To control whether the containers is out or still in. It is mrn_type array.
	int_array ccavectout2_list: Array of the containers' ids that is exited by the agent. Same as above.
	int_array cca_list: *mrn type array.*
Functions	***CustomClearanceAgent_receive_a3:*** The CCA receives A3 document
	CustomClearanceAgent_send_customsdeclaration: The customs clearance agent sends the Customs Declaration.
	CustomClearanceAgent_receives_clearcode_pickup_del_ord: The customs clearance agent receives Pick up, the Authorization Code and the Delivery Order for the goods.
	CustomClearanceAgent_require_transportation: The customs clearance agent requires the Order of Transportation. The function that starts transportation process.
	CustomClearanceAgent_receive_confirmation_transportation: The customs clearance agent receives confirmation of transport.
	CustomClearanceAgent_send_leaving_documentation: The customs clearance agent sends the documentation to be able to withdraw and exit. CCA send it to the Gate.
	CustomClearanceAgent_receive_permission_exit: The customs clearance agent receives the permission exit from the gate.
	CustomClearanceAgent_book_exit: The customs clearance agent books for the exit.
Messages	***customclearanceagent_customsdeclaration_customs:*** contains MRN. Customs Declaration to Customs.
	customclearanceagent_bookforpickup_terminal: Book for pick up.
	customclearanceagent_ot_request_deliveryservice: Order of transport.
	customclearanceagent_doc_gate: leaving documents to the Gate.
	customclearanceagent_w_gate: books for exit document.

Table 3 CCA's Variables, Functions and Messages

2.2.4. Variables, Functions and Messages of Delivery Service Agent

DELIVERY SERVICE	
Data Types	***deliveryservicevectormrn_array:*** mrn_type array.
Memory Variables	***int transportationready:*** Document that indicates the container can be transported.
	access_type access: A data type that indicates access to the agent, consists of int id and int value.
	gateout_type gateout: Data type indicates the exit from the agent, it consists of int id and int value.
	mrn_type_array deliveryservicevectormrn_list: Array for the container MRNs within the agent.
	mrn_type_array vectormrn_list. Similar to the one above.
	mrn_type mrntemp_deliveryservice: Temporary MRN document.
	int sizeend: Number of containers in the agent.
	int_array dsvectin_list: Array for the containers' ids that entered into the agent.
	int_array dsvectin2_list: Array for the containers' ids entered into the agent. ***int_array dsvectout_list:*** Array of the container ids exited by the agent.
	int_array dsvectout2_list: Array of the container ids exited by the agent.
Functions	***DeliveryService_receive_request_transportation:*** receives request for transportation from CCA.
	DeliveryService_request_transportation_to_deliverer: asks for transportation to deliverer.

	DeliveryService_receive_confirm_transportation: receive confirmation for transportation from Deliverer.
	DeliveryService_send_confirm_transportation_to_cca: send confirmaiton to CCA for the trasportation.
Messages	***deliveryservice_ot_request_deliverer*:** request Order of transport from Deliverer.Contains MRN.
	deliveryservice_ot_customclearanceagent: send Order of Transport to CCA.

Table 4 Deliver Service Agent's Variables, Functions and Messages

2.2.5. Variables, Functions and Messages of Deliverer Agent

DELIVERER	
Data Types	***deliverervectormrn_array:*** array for MRN type in Deliverer.
Memory Variables	***int_array mrn_vector_list:*** Array of the container ids in the agent.
	access_type access: A data type that indicates access to the agentand it consists of int id and int value.
	gateout_type gateout: Data type indicates the exit from the agentand it consists of int id and int value.
	mrn_type_array deliverervectormrn_list: Array of the MRNs of the containers within the agent.
	mrn_type_array vectormrn_list: Similar to the one above.
	mrn_type mrntemp_deliverer: Temporary MRN document.
	int sizeend: Number of containers in the agent.
	int_array deliverervectin_list: Array for the ids of the containers that entered into the agent.
	int_array deliverervectin2_list: Array for the ids of the containers that entered into the agent.
	int_array deliverervectout_list: Array of the container ids that is exited by the agent.

	int_array deliverervectout2_list: Array of the container ids that is exited by the agent.
Functions	***Deliverer_receive_request_transport:*** request to transport.
	Deliverer_confirm_transportation_to_deliveryservice: confirm transportation to Delivery service.
	Deliverer_receive_doc: receives document from Gate for exit.
	Deliverer_receive_date: date to exit from Gate.
	Deliverer_request_access: requires access to Terminal.
	Deliverer_enter_terminal: enters to Terminal if it is allowed.
	Deliverer_leaveterminal: Deliverer leaves from Terminal.
Messages	***deliverer_ot_deliveryservice:*** Order of Transport. Contains MRN.
	deliverer_del_ord_terminal: Delivery Order.
	deliverer_enter_area: enter Area.

Table 5 Deliverer Agent's Variables, Functions and Messages

2.2.6. Variables, Functions and Messages of Terminal Agent

TERMINAL	
Data Types	***terminalvectormrn_array:*** array MRN in Terminal.
	pickup_type: pick up ID and value.
	gateout_type: gateout ID and value.
Memory Variables	***int_array mrn_vector_list:*** Array of the container ids in the agent.
	int pickupvalue: Document indicates that the container can be picked up.
	int accessvalue: Document indicates that the container can access the terminal.

	pickup_type pickup: Data type defines that the container can be picked up, consists of int id and int value.
	access_type access: Data type defines that the container can access the terminal, consists of int id and int value.
	gateout_type gateout: Data type indicates that the container can exit the terminal, consists of int id and int value.
	int Container: Variable indicats that the container has been picked up.
	int loading_container: Variable indicates that the container is being loaded.
	int finish_loading_container: Variable indicats that the loading of the container has ended.
	mrn_type_array terminalvectormrn_list: Array of the MRNs of the containers within the agent.
	mrn_type_array vectormrn_list: Similar to the one above.
	mrn_type mrntemp_terminal: Temporary MRN document.
	int sizeend: Number of containers in the agent.
	int_array terminalvectin_list: Array of container ids entered into the agent.
	int_array terminalvectin2_list: Array of container ids entered into the agent.
	int_array terminalvectout_list: Array of the container ids exited by the agent.
	int_array terminalvectout2_list: Array of the container ids exited by the agent.
Functions	***Terminal_receive_request_for_pickup:*** The terminal receives a pick-up reservation from the customs clearance agent.
	Terminal_allow_pickup: The terminal authorizes for the pick up.
	Terminal_receive_request_access: receives request access to Terminal. The terminal receives the request for access from the deliverer.

	Terminal_elaborate_request: Terminal elaborates request.
	Terminal_allow_gate_in: The terminal authorizes the deliverer to enter.
	Terminal_find_container: The terminal finds the container and places it in the area.
	Terminal_load_goods: The terminal authorizes for loading of goods.
	Terminal_finish_loading: The terminal receives communication that the container is loaded.
	Terminal_gateout: The terminal authorizes the deliverer to collect the container.
Messages	***terminal_pickup_customclearanceagent:*** container out to Terminal.
	terminal_gatein_deliverer: send gate in to Deliverer.
	terminal_gateout_deliverer: gate out to Deliverer.
	terminal_loading_area: loading to Area.

Table 6 Terminal Agent's Variables, Functions and Messages

2.2.7. Variables, Functions and Messages Finance Police

FINANCE POLICE	
Data Types	***financepolicevectormrn_array:*** array MRN in FP.
Memory Variables	***int_array mrn_vector_list:*** Array of the container ids in the agent.
	mrn_type_array financepolicevectormrn_list: Array of MRNs of the containers within the agent. ***mrn_type_array vectormrn_list:*** Similar to the one above.
	mrn_type mrntemp_fp: Temporary MRN document.
	int sizeend: Number of containers in the agent.

	int_array fpvectin_list: Array of the container ids entered into the agent.
	int_array fpvectin2_list: Array of the container ids entered into the agent.
	int_array fpvectout_list: Array of the container ids exited by the agent.
	int_array fpvectout2_list: Array of the container ids exited by the agent.
Functions	***Financepolice_receive_documentation:*** receives Security Clearance from Gate.
	Financepolice_give_nullaosta: gives Security Clearance if the goods are clear.
Messages	***financepolice_no_gate:*** Security Clearance document from FP to Gate. It contains MRN.

Table 7 Finance Police Agent's Variables, Functions and Messages

2.2.8. Variables, Functions and Messages of Gate Agent

GATE	
Data Types	***gatevectormrn_array:*** array MRN in Gate.
Memory Variables	***int permissionexitvalue:*** Document indicates that the container can exit.
	dv_type dv: Data type that indicates the vector document for the agent, consists of int id and int value.
	mrn_type_array gatevectormrn_list: Array of the MRNs of the containers within the agent.
	mrn_type mrntemp_gate: Temporary MRN document.
	int sizeend: Number of containers in the agent.
	int_array gatevectin_list: Array of the container ids entered into the agent.

	int_array gatevectin2_list: Array of the container ids entered into the agent.
	int_array gatevectin2_list: Array of the container ids entered into the agent.
	int_array gatevectout_list: Array of the container ids exited by the agent.
	int_array gatevectout2_list: Array of the container ids exited by the agent.
Functions	***Gate_receive_doc_for_exit:*** Receives leaving document from CCA.
	Gate_allow_exit: allow permision exit to CCA.
	Gate_receive_book_exit: Receives booking documents for exit from CCA.
	Gate_send_documentation: sends document to Deliverer.
	Gate_communicate_date_exit: communicates with Deliverer for exit date.
	Gate_request_nullaosta: Requests Security Clearance to FP.
	Gate_receive_no: Receives Security Clearance from FP.
	Gate_send_no: send Security Clearance to Deliverer.
Messages	***gate_permissionexit_customclearanceagent:*** permission for exit.
	gate_doc_deliverer: document for exit.
	gate_d_deliverer: date for exit.
	gate_no_request_financepolice: Security Clearance request from FP.
	gate_no_deliverer: Security Clearance to Deliverer.

Table 8 Gate Agent's Variables, Functions and Messages

2.2.9. Variables, Functions and Messages of Area Agent

AREA	
Data Types	***areavectormrn_array:*** array MRN in Area.
Memory Variables	***mrn_type_array areavectormrn_list:*** Array of the MRNs of the containers within the agent.
	mrn_type_array vectormrn_list: Similar to the one above.
	int_array mrn_vector_list: Array of container ids.
	mrn_type mrntemp_area: Temporary MRN document.
	int sizeend: Number of containers in the agent.
	int_array areavectin_list: Array of the container ids entered into the agent.
	int_array areavectin2_list: Array of the container ids entered into the agent.
	int_array areavectout_list: Array of the container ids exited by the agent.
	int_array areavectout2_list: Array of the container ids exited by the agent.
Functions	***Area_add_truck:*** Deliverer adds truck in Area if there is a space.
	Area_load_truck: Deliverer loads Truck.
Messages	***area_finishloading_terminal:*** Finish loading.

Table 9 Area Agent's Variables, Functions and Messages

2.3. Execution of the Model on FLAME

The FLAME framework contains an analysis program called "xparser" which analyzes the model in XML and transforms it into the source code of the simulation program, compiling it together with the implementation codes of the functions of the agents and the template files. This is the "parsing" procedure, in the figure 5 above on the FLAME part.

This procedure generates the following documentation:

- stategraph.dot - an acyclic graph of states, it includes functions and messages among agents in the model.
- stategraph_colour.dot - as above but functions are colored.
- process_order_graph.dot - as above, but message synchronization is shown.
- latex.dot - the description of the model written in a LaTeX document.

It also generates the following files related to the source code of the simulation program:

- Makefile - a compilation script used by the "make" program.
- xml.c - the source code that manages the simulation inputs and outputs.
- main.c - the source code that contains the loop of the main program.
- header.h - a header file in C language that contains the global variables and the declarations of the all functions in source code.
- memory.c - the source code that manages the memory requirements of the simulation.
- low_primes.h - contains data used for partitioning agents.
- messageboards.c - the source code that manages the functions related to the messages, it means message functionality.
- partitioning.c - the source code file that manages the partitioning of agents between nodes in parallel.
- timing.c - the source code that provides temporal routines.
- Doxyfile - a configuration file to generate documentation using the "Doxygen" program.
- rules.c - the code contains the rules generated for the conditions of the functions and the filters for input messages.

For each type of agent an associated header file is also created:

• <agent_name> _agent_header.h - the header file that contains the macros necessary to access the variables in the agent's memory.

To do this, the GXParser software is used on the Windows 8.1 and Windows 10.

In the model, the software package is used that includes;

- Xparser GUI : Parser for FLAME models;
- GNU GCC compiler : C compiler for model + framework code;
- Flame Editor : Generate model.xml file, XML description of model;
- Population GUI : Generates initialization files, population description. (to create an initial state as input of a simulation);
- Simulation GUI : Settings for simulation experiments and data analysis.

It is worth saying that *GCC compiler*, *XParser*(Model descripton-XML)and *LIBMBOARD* (Agent communication and behaviour implementations-C functions) are required software components.
Furthermore, there are also Optional software components. They are PopGUI, ExpGUI as Initialization and experiment setup, VisGUI as Data Visualiser.

2.3.1. GXParser

First, XParser GUI is used. Xparser generates simulation program of FLAME. It is a set of compilation files that is possible to compile the files with the developers' files to produce a simulation package for the execution of the simulations. The files that Xparser needs during rhe running,

- Model.xml – It is not a single files.It is a series of xml files that includes the all structure of the model like agents' descriptions, memory variables, agents' functions and messages.
- Functions.c – they are multiple '.c' files that include the implementations of the agent functions. These are specified in the xml files.
- 0.xml - The initial states of the memory variables of the agents are represented in this file such as all paremeters are initalized here. The number of the resulting XML files depends on the number of iterations specified to run a model (through Main.exe) [58].

The processes of the Xparser;

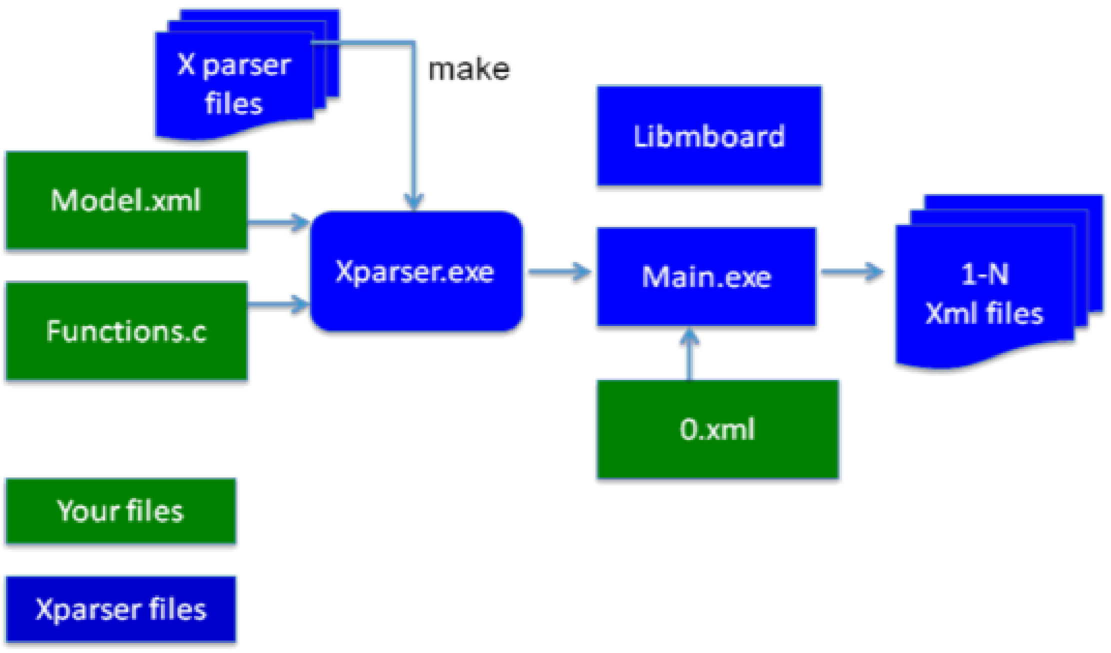
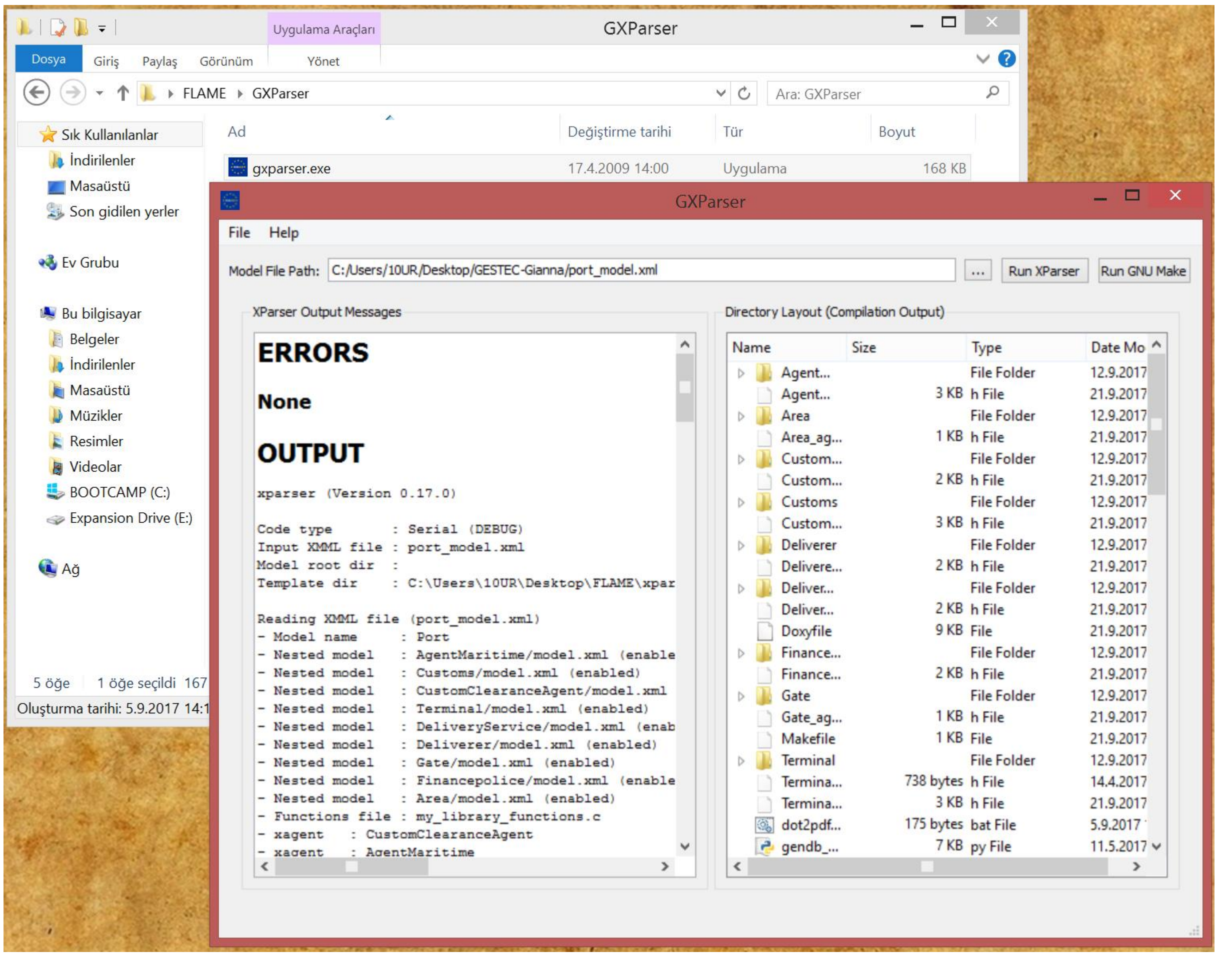


Illustration 2 Block Diagram of XParser

Block diagram of the Xparser, the FLAME simulation program. Blocks in blue are the files automatically generated. The green blocks are modeler files.

To this program as input only the main XML model is needed, located inside the folder that contains the other models related to it and the C codes of the functions: first of all the Xparser function will be launched, i.e. the one explained above, which will check the model and produce all the files; therefore, if there are no errors in the model, we will launch the GNU Make function, which is the compiler of the C code generated by XParser. If there are no syntax or compile errors in the functions of the agents the executable will be created, i.e. the file "main.exe".

You can see an image of the application window in Figure 20. On the left of the window, it can be seen the Errors of the model.xml files occurred on the structure and there is also output section that includes the name of the model, agents as nested models, function files etc. On the right side of the window, there is directory layout that has all the files about the model that is created.



Ad	Değiştirme tarihi	Tür	Boyut
gxparser.exe	17.4.2009 14:00	Uygulama	168 KB

Model File Path: C:/Users/10UR/Desktop/GESTEC-Gianna/port_model.xml

```
ERRORS

None

OUTPUT

xparser (Version 0.17.0)

Code type       : Serial (DEBUG)
Input XMML file : port_model.xml
Model root dir  :
Template dir    : C:\Users\10UR\Desktop\FLAME\xpar

Reading XMML file (port_model.xml)
- Model name      : Port
- Nested model    : AgentMaritime/model.xml (enable
- Nested model    : Customs/model.xml (enabled)
- Nested model    : CustomClearanceAgent/model.xml
- Nested model    : Terminal/model.xml (enabled)
- Nested model    : DeliveryService/model.xml (enab
- Nested model    : Deliverer/model.xml (enabled)
- Nested model    : Gate/model.xml (enabled)
- Nested model    : Financepolice/model.xml (enable
- Nested model    : Area/model.xml (enabled)
- Functions file : my_library_functions.c
- xagent    : CustomClearanceAgent
- xagent    : AgentMaritime
```

Name	Size	Type	Date Mo
Agent...		File Folder	12.9.2017
Agent...	3 KB	h File	21.9.2017
Area		File Folder	12.9.2017
Area_ag...	1 KB	h File	21.9.2017
Custom...		File Folder	12.9.2017
Custom...	2 KB	h File	21.9.2017
Customs		File Folder	12.9.2017
Custom...	3 KB	h File	21.9.2017
Deliverer		File Folder	12.9.2017
Delivere...	2 KB	h File	21.9.2017
Deliver...		File Folder	12.9.2017
Deliver...	2 KB	h File	21.9.2017
Doxyfile	9 KB	File	21.9.2017
Finance...		File Folder	12.9.2017
Finance...	2 KB	h File	21.9.2017
Gate		File Folder	12.9.2017
Gate_ag...	1 KB	h File	21.9.2017
Makefile	1 KB	File	21.9.2017
Terminal		File Folder	12.9.2017
Termina...	738 bytes	h File	14.4.2017
Termina...	3 KB	h File	21.9.2017
dot2pdf...	175 bytes	bat File	5.9.2017
gendb_...	7 KB	py File	11.5.2017

Figure 20 Screen of the GXParser program with the output of the XParser function

2.3.2. PopGUI

In order to launch simulations, initialization file of the model is needed. This initialization file is called '0.xml' that is a template file. At that point, 0.xml is created by PopGUI.When the XML template is ready and given then if the executable code is compiled, PopGUI allows to set multiple varibles.

Firstly, it is necessary to decide on the *Properties* Section, how many Regions of the model will be used. For example, how many parallel simulations of the same model will be initialized. It is worth saying that one simulation is used only because simulation parallelization option can only be used on high performance computer. The regions can be called with a name that is wanted by developer.

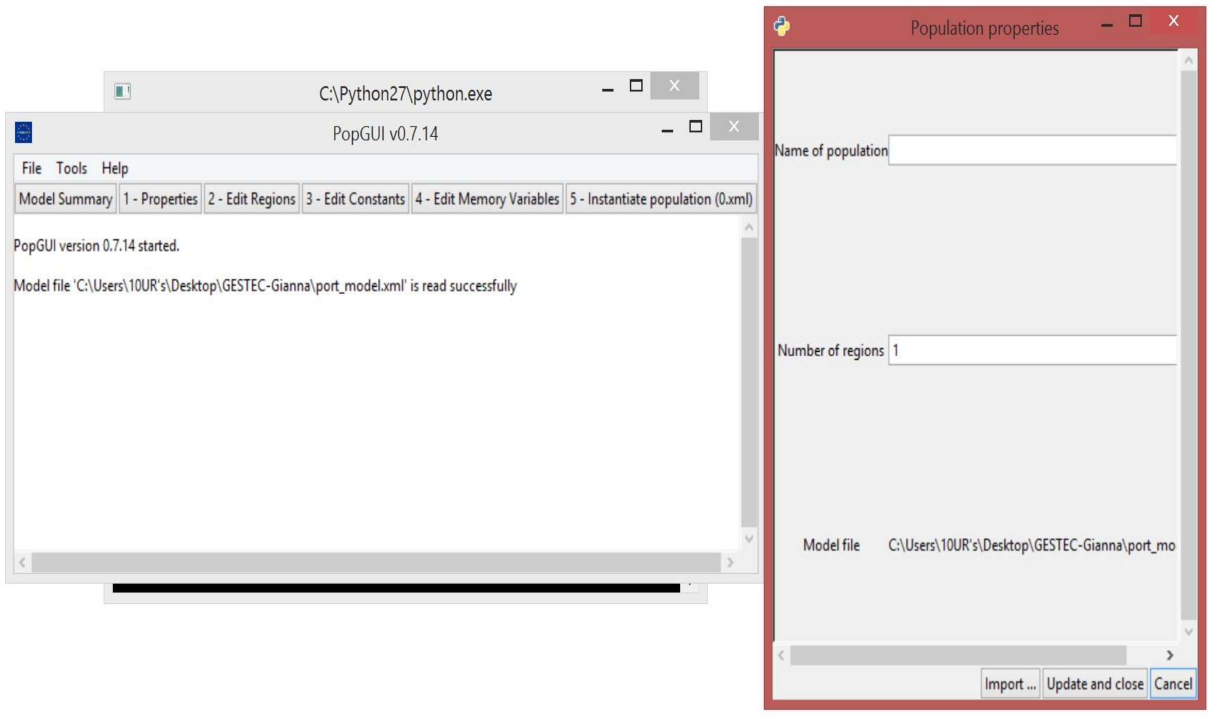
Furthermore, for each region, it is necessary to decide,which and how many agents will be used. These selection will be chosen through the *Edit Regions* section.

The values of the environmental constants can be edited on the *Edit Constant* section and as a last, the values of the agent variables can be chosen on the *Edit Memory Variables* section.

In particular, it is mandatory to change ens_max in Agent Maritime and Customs_work_max in Customs Agent (They are all described in the modelling section), setting them different than 0, and if desired, environmental constants, such as debug_mode, means debugging function, can be activated.

After all parameters that are wanted to choose, are edited, initialization file 0.xml and its' population file 0.pop can be saved.

Figure 21 shows the Properties section that Name of the population and Number of regions can be chosen.



```
PopGUI version 0.7.14 started.

Model file 'C:\Users\10UR's\Desktop\GESTEC-Gianna\port_model.xml' is read successfully
```

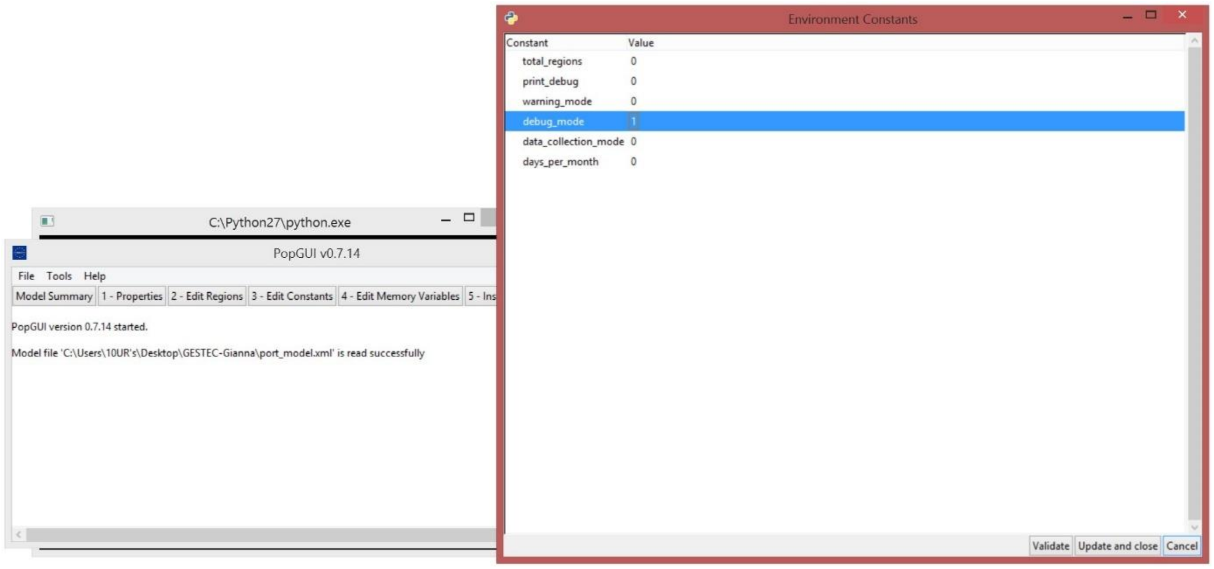
Name of population

Number of regions 1

Model file C:\Users\10UR's\Desktop\GESTEC-Gianna\port_mo

Figure 21 PopGUI Properties section

Figure 22 shows us the Environmental Constants that can be edited. For example, for activation of the debugging functions that is fitted on the model can be chosen as 0 and 1. These represent enable or disable of the file. If it is 1, debug mode is active.


Environment Constants
C:\Python27\python.exe
PopGUI v0.7.14
File Tools Help
Model Summary | 1 - Properties | 2 - Edit Regions | 3 - Edit Constants | 4 - Edit Memory Variables | 5 - Ins
Validate | Update and close | Cancel

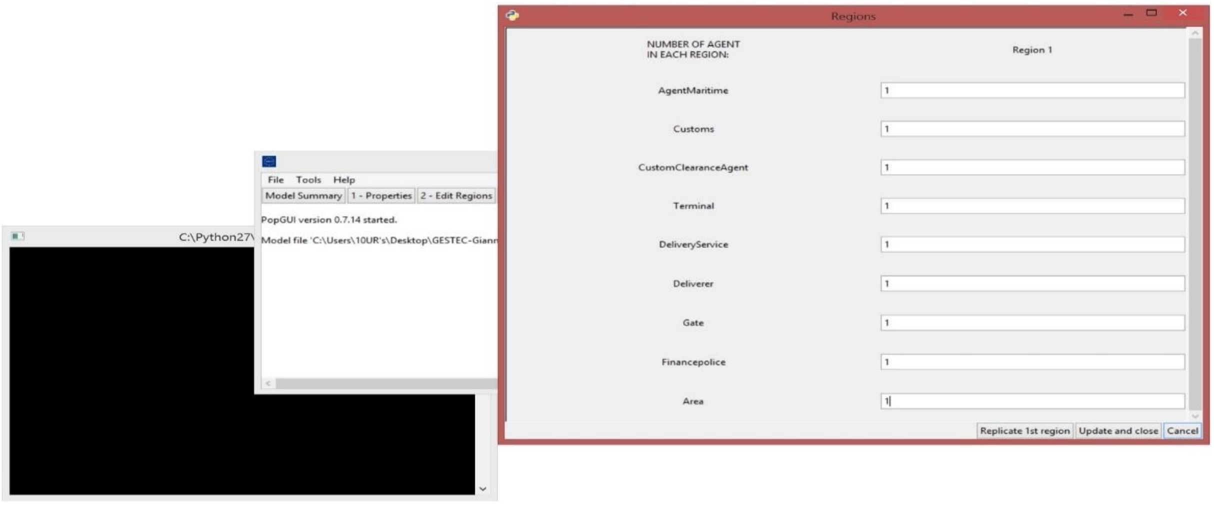
Constant	Value
total_regions	0
print_debug	0
warning_mode	0
debug_mode	1
data_collection_mode	0
days_per_month	0

```
PopGUI version 0.7.14 started.
Model file 'C:\Users\10UR's\Desktop\GESTEC-Gianna\port_model.xml' is read successfully
```

Figure 22 PopGUI Environmental Constants

‘Port_model.xml’, ‘init.xml’ , and ‘population(.pop)’ files are used during the calibration process.

It is worth to mentioning that firstly the model (.xml) have to be identified in the POPGui.


Regions
C:\Python27\
File Tools Help
Model Summary | 1 - Properties | 2 - Edit Regions
Replicate 1st region | Update and close | Cancel

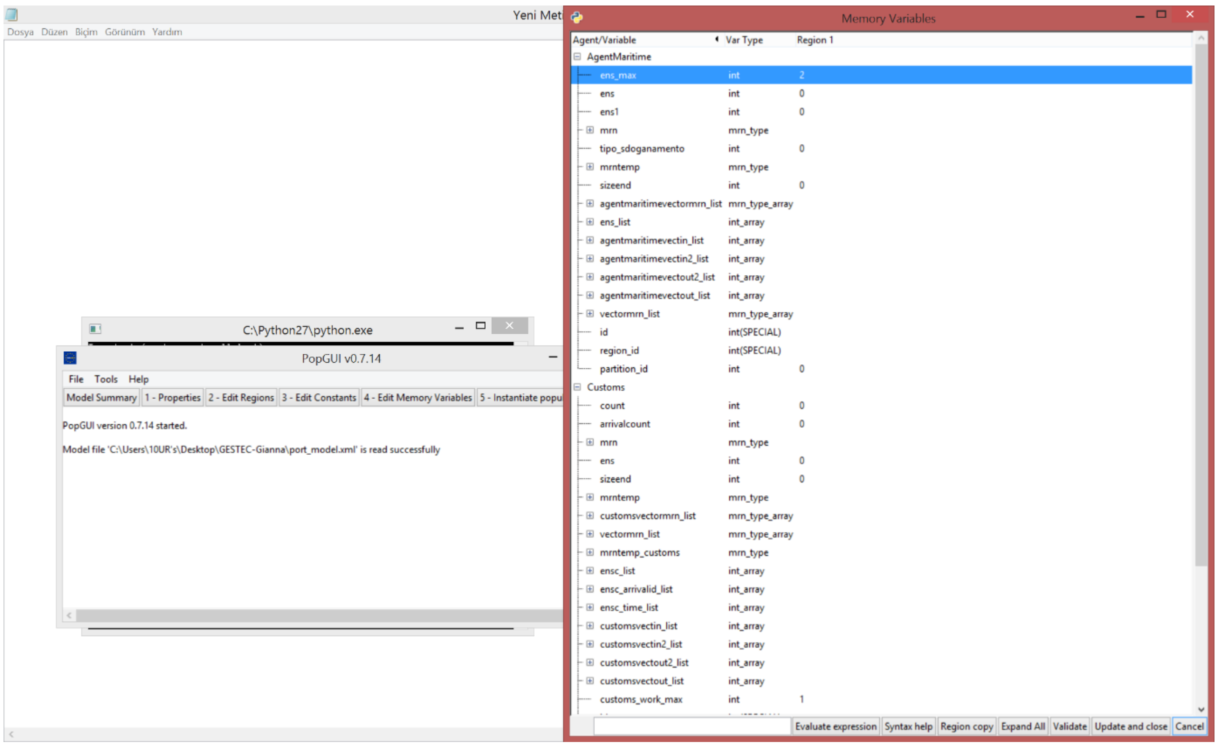
```
PopGUI version 0.7.14 started.
Model file 'C:\Users\10UR's\Desktop\GESTEC-Giann
```

NUMBER OF AGENT IN EACH REGION:	Region 1
AgentMaritime	1
Customs	1
CustomClearanceAgent	1
Terminal	1
DeliveryService	1
Deliverer	1
Gate	1
Financepolice	1
Area	1

Figure 23 PopGUI - Edit Regions

As it can be seen in Figure 23, all the agents that model includes are in the *Regions* tab of the POPGui. For the calibration phase, all agents' regions are edited as 1. So there are nine agents and these nine agents have one region.

In another tab named *Edit Memory Variables*, the value of the variables can be changed. In the Figure 24, memory variables can be seen respectively and the values can be updated.



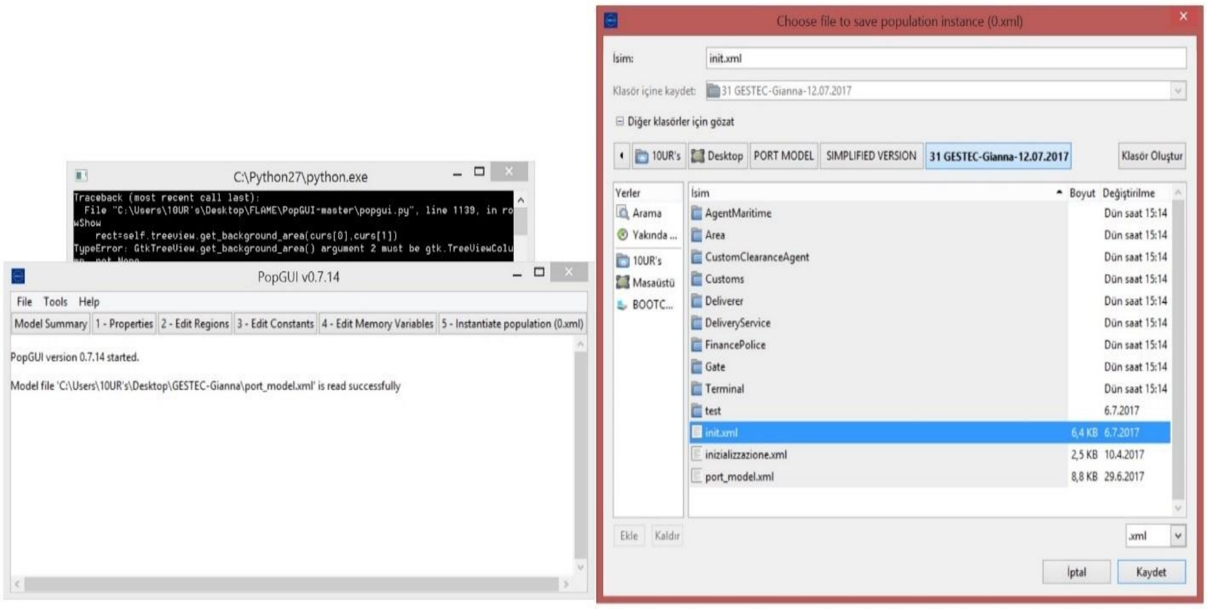
```
PopGUI version 0.7.14 started.

Model file 'C:\Users\10UR's\Desktop\GESTEC-Gianna\port_model.xml' is read successfully
```

Agent/Variable	Var Type	Region 1
AgentMaritime		
ens_max	int	2
ens	int	0
ens1	int	0
mrn	mrn_type	
tipo_sdoganamento	int	0
mrntemp	mrn_type	
sizeend	int	0
agentmaritimevectormrn_list	mrn_type_array	
ens_list	int_array	
agentmaritimevectin_list	int_array	
agentmaritimevectin2_list	int_array	
agentmaritimevectout2_list	int_array	
agentmaritimevectout_list	int_array	
vectormrn_list	mrn_type_array	
id	int(SPECIAL)	
region_id	int(SPECIAL)	
partition_id	int	0
Customs		
count	int	0
arrivalcount	int	0
mrn	mrn_type	
ens	int	0
sizeend	int	0
mrntemp	mrn_type	
customsvectormrn_list	mrn_type_array	
vectormrn_list	mrn_type_array	
mrntemp_customs	mrn_type	
ensc_list	int_array	
ensc_arrivalid_list	int_array	
ensc_time_list	int_array	
customsvectin_list	int_array	
customsvectin2_list	int_array	
customsvectout2_list	int_array	
customsvectout_list	int_array	
customs_work_max	int	1

Figure 24 POPGui-Edit Memory Variables

The most important information for this tab is; all the agents and all the memory variables inside the agents can be seen here directly. In the case of port model, 'ens_max' and 'work_max' memory variables is changed for getting different results in different conditions. The main idea is to see bottleneck and to watch gateins and gateouts of the containers. Ens_max is the maximum number of containers that AgentMaritime can elaborate in the same iteration (time) and customs_work_max is the maximum number of containers that agent Customs can work at the same time. These two variables help us to show if there are bottleneck or not and the size of the containers, their attitudes, gatein times, gateout times etc.

Furthermore, before starting the simulation, it needs two files. One is '0.xml (init.xml)' and 'main.exe'. Main.exe is created by Xparser and init.xml is created by PopGUI. That is why PopGUI is really important section in the simulation process. In the Instantiate population tab, all changed values can be saved as what have done in regions, constants or variables. The tab can be seen in Figure 25.



```
Traceback (most recent call last):
  File "C:\Users\10UR's\Desktop\FLAME\PopGUI-master\popgui.py", line 1139, in ro
wShow
    rect=self.treeview.get_background_area(curs[0],curs[1])
TypeError: GtkTreeView.get_background_area() argument 2 must be gtk.TreeViewColu
mn, not None
```

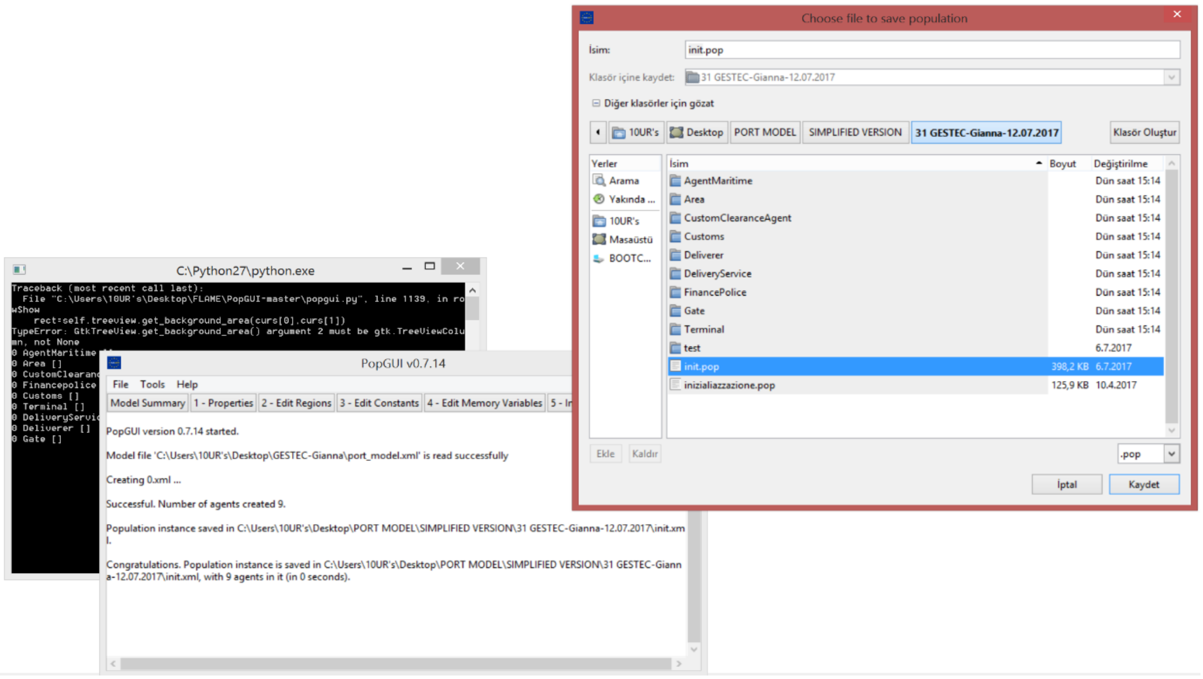
```
PopGUI version 0.7.14 started.

Model file 'C:\Users\10UR's\Desktop\GESTEC-Gianna\port_model.xml' is read successfully
```

C:\Python27\python.exe
PopGUI v0.7.14
File Tools Help
Model Summary | 1 - Properties | 2 - Edit Regions | 3 - Edit Constants | 4 - Edit Memory Variables | 5 - Instantiate population (0.xml)
Choose file to save population instance (0.xml)
İsim: init.xml
Klasör içine kaydet: 31 GESTEC-Gianna-12.07.2017
Diğer klasörler için gözat
10UR's | Desktop | PORT MODEL | SIMPLIFIED VERSION | 31 GESTEC-Gianna-12.07.2017
Klasör Oluştur
Yerler: Arama, Yakında ..., 10UR's, Masaüstü, BOOTC...
Ekle Kaldır
.xml
İptal Kaydet

İsim	Boyut	Değiştirilme
AgentMaritime		Dün saat 15:14
Area		Dün saat 15:14
CustomClearanceAgent		Dün saat 15:14
Customs		Dün saat 15:14
Deliverer		Dün saat 15:14
DeliveryService		Dün saat 15:14
FinancePolice		Dün saat 15:14
Gate		Dün saat 15:14
Terminal		Dün saat 15:14
test		6.7.2017
init.xml	6,4 KB	6.7.2017
inizializzazione.xml	2,5 KB	10.4.2017
port_model.xml	8,8 KB	29.6.2017

Figure 25 PopGUI-Instantiate population

Last phase of the PopGUI is saving the '.pop' file. The population file have to be saved as '0.pop' or 'init.pop'. It is shown in Figure 26.



```
Traceback (most recent call last):
  File "C:\Users\10UR's\Desktop\FLAME\PopGUI-master\popgui.py", line 1139, in ro
wShow
    rect=self.treeview.get_background_area(curs[0],curs[1])
TypeError: GtkTreeView.get_background_area() argument 2 must be gtk.TreeViewColu
mn, not None
0 AgentMaritime
0 Area []
0 CustomClearanc
0 Financepolice
0 Customs []
0 Terminal []
0 DeliveryServic
0 Deliverer []
0 Gate []
```

```
PopGUI version 0.7.14 started.

Model file 'C:\Users\10UR's\Desktop\GESTEC-Gianna\port_model.xml' is read successfully

Creating 0.xml ...

Successful. Number of agents created 9.

Population instance saved in C:\Users\10UR's\Desktop\PORT MODEL\SIMPLIFIED VERSION\31 GESTEC-Gianna-12.07.2017\init.xm
l.

Congratulations. Population instance is saved in C:\Users\10UR's\Desktop\PORT MODEL\SIMPLIFIED VERSION\31 GESTEC-Giann
a-12.07.2017\init.xml, with 9 agents in it (in 0 seconds).
```

C:\Python27\python.exe
PopGUI v0.7.14
File Tools Help
Model Summary | 1 - Properties | 2 - Edit Regions | 3 - Edit Constants | 4 - Edit Memory Variables | 5 - In
Choose file to save population
İsim: init.pop
Klasör içine kaydet: 31 GESTEC-Gianna-12.07.2017
Diğer klasörler için gözat
10UR's | Desktop | PORT MODEL | SIMPLIFIED VERSION | 31 GESTEC-Gianna-12.07.2017
Klasör Oluştur
Yerler: Arama, Yakında ..., 10UR's, Masaüstü, BOOTC...
Ekle Kaldır
.pop
İptal Kaydet

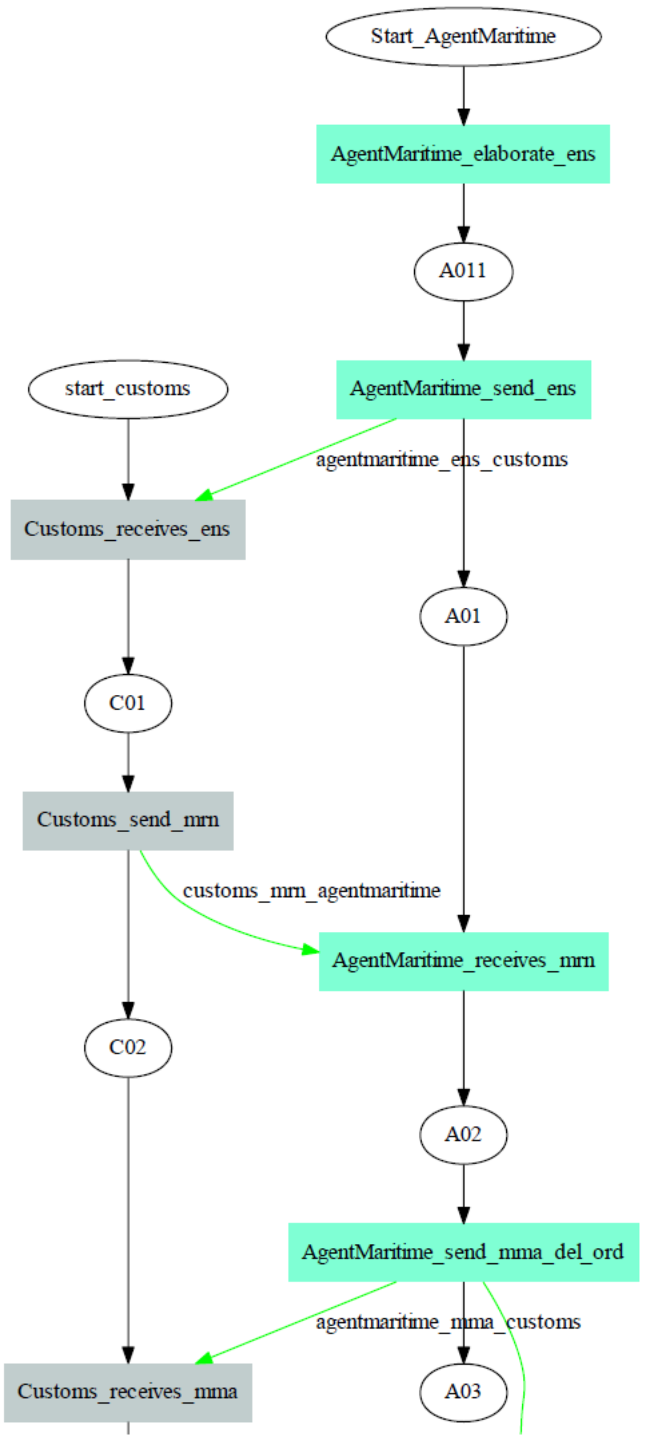
İsim	Boyut	Değiştirilme
AgentMaritime		Dün saat 15:14
Area		Dün saat 15:14
CustomClearanceAgent		Dün saat 15:14
Customs		Dün saat 15:14
Deliverer		Dün saat 15:14
DeliveryService		Dün saat 15:14
FinancePolice		Dün saat 15:14
Gate		Dün saat 15:14
Terminal		Dün saat 15:14
test		6.7.2017
init.pop	398,2 KB	6.7.2017
inizializzazione.pop	125,9 KB	10.4.2017

Figure 26 PopGUI-save

After the phase of population and the change of values, launching simulations are ready.

2.3.3. Stategraph

The stategraph explains the model in depth and it is created through the GXParser program. Stategraph is a flowchart of the model that contains states, functions and messages. A small part of the stategrapg is shown in Figure 27.



Start_AgentMaritime
AgentMaritime_elaborate_ens
A011
start_customs
AgentMaritime_send_ens
agentmaritime_ens_customs
Customs_receives_ens
A01
C01
Customs_send_mrn
customs_mrn_agentmaritime
AgentMaritime_receives_mrn
C02
A02
AgentMaritime_send_mma_del_ord
agentmaritime_mma_customs
Customs_receives_mma
A03

Figure 27 Stategrapgh of the model

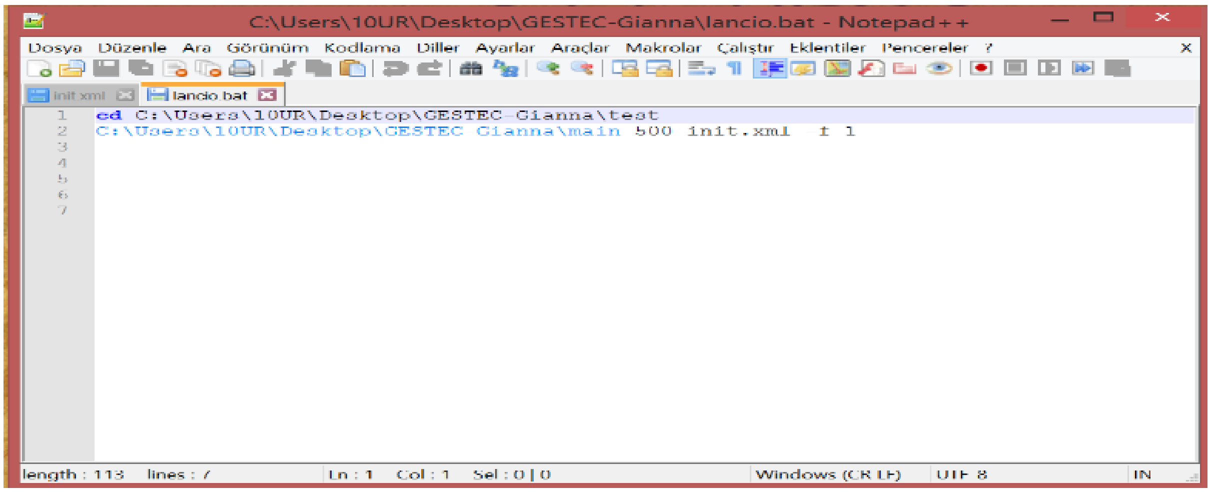
As it is seen in Figure 27, the beginning of Agent Maritime and Customs agent diagrams. Ovals are the states with the name inside them (as it is seen the first one is always defined as "start_agentename" while the following have some abbreviations in numerical order), the colored rectangles are the functions with the name inside (a different color for each agent), the black arrows indicate the flow of the agent's behavior from state to function and vice versa (and in case of duplication of the path , the single arrow will have indicated the condition) while the green arrows indicate the messages, which start from the sender function and arrive at the recipient function and have the close tag indicating that the name of the message (in the image for example it is seen the message agentmaritime_ens_customs).

2.3.4. Launching simulations and read results

At this point, the simulations process and executed executable code are explained.

The file named launch.bat has been produced in which the path of the test folder and its' name, the name of the initialization file (init.xml file is used that is created by default but it can be changed), the number of iteration wanted and the frequency of output creation (which are structured XML files equal to the initialization file but that are generated in the simulation with the values).

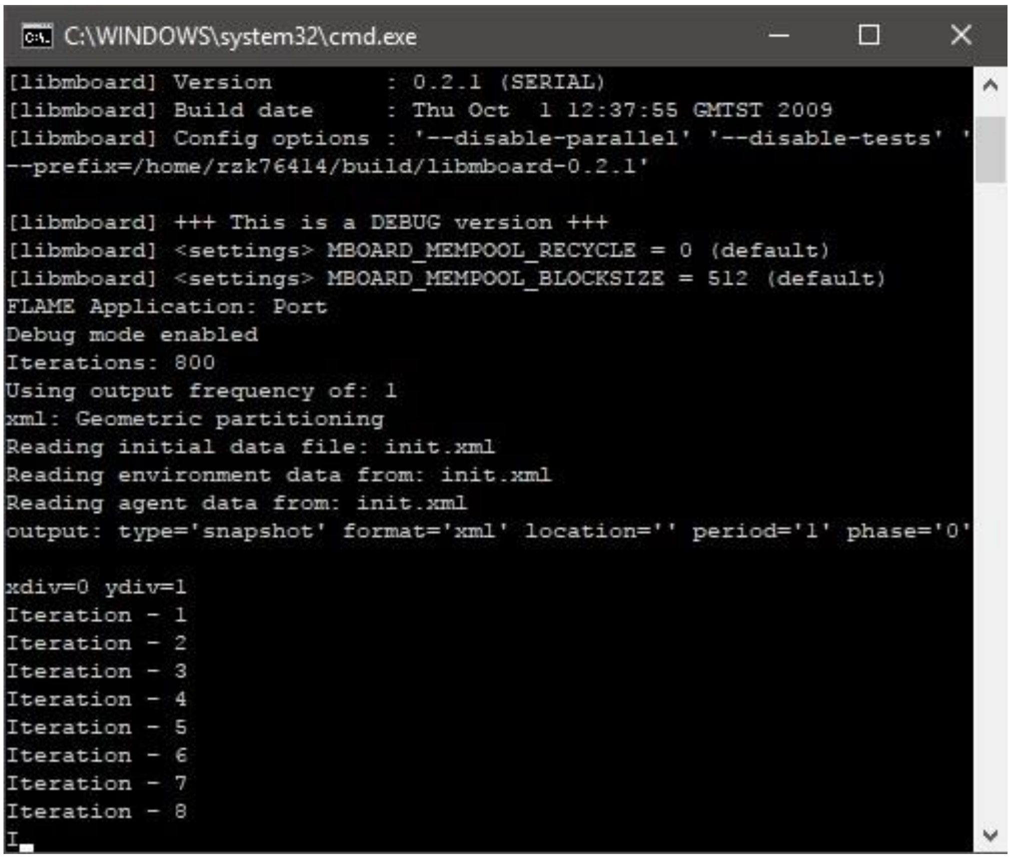
After modifying it depending on developer's needs, or inserting the parameters described above in Figure 28 as developer's wants, just launch it as a normal application and the screen contains the progress of the simulation is shown in Figure 29 which will close as soon as it is finished.


C:\Users\10UR\Desktop\GESTEC-Gianna\lancio.bat - Notepad++
Dosya Düzenle Ara Görünüm Kodlama Diller Ayarlar Araçlar Makrolar Çalıştır Eklentiler Pencereler ?
init.xml
lancio.bat
length : 113 lines : 7 Ln : 1 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 IN

```
cd C:\Users\10UR\Desktop\GESTEC-Gianna\test
C:\Users\10UR\Desktop\GESTEC-Gianna\main 500 init.xml -f 1
```

Figure 28 Simulation parameters – Launch.bat with 500 iterations and frequency 1

Figure 28 shows the launch.bat file and the code inside. On this launch.bat, the iteration is chosen 500 and frequency is 1.



```
C:\WINDOWS\system32\cmd.exe
[libmboard] Version       : 0.2.1 (SERIAL)
[libmboard] Build date    : Thu Oct  1 12:37:55 GMTST 2009
[libmboard] Config options : '--disable-parallel' '--disable-tests' '
--prefix=/home/rzk76414/build/libmboard-0.2.1'

[libmboard] +++ This is a DEBUG version +++
[libmboard] <settings> MBOARD_MEMPOOL_RECYCLE = 0 (default)
[libmboard] <settings> MBOARD_MEMPOOL_BLOCKSIZE = 512 (default)
FLAME Application: Port
Debug mode enabled
Iterations: 800
Using output frequency of: 1
xml: Geometric partitioning
Reading initial data file: init.xml
Reading environment data from: init.xml
Reading agent data from: init.xml
output: type='snapshot' format='xml' location='' period='1' phase='0'

xdiv=0 ydiv=1
Iteration - 1
Iteration - 2
Iteration - 3
Iteration - 4
Iteration - 5
Iteration - 6
Iteration - 7
Iteration - 8
I_
```

Figure 29 Start screen of simulation (execution)

The model will have produced, if the environmental constant debug_mode is set to 1 on the PopGUI, the TXT files generated by the functions inside the code and the output XML, which will be in the test folder otherwise they will be only XML products.

The obtained data can be stored in a database (a file '.db' format) to use them more easily that can be opened with software like SQLite.

TXTs have been used in the early stages of model development as a debugging tool, as it is possible to produce one for each function.

On the Port Model, the MATLAB is used to produce graphics and analyze them as MATLAB is one of the most famous and used mathematical data processing program on the scientific area. The following programs have databases as input, from which we extract data through queries in a language similar to SQL, and various types of graphs are produced as an output.

2.3.5. Virtual Appliance

A parenthetical remark have to be for the Virtual Machines (VM) [7], [8], [33].

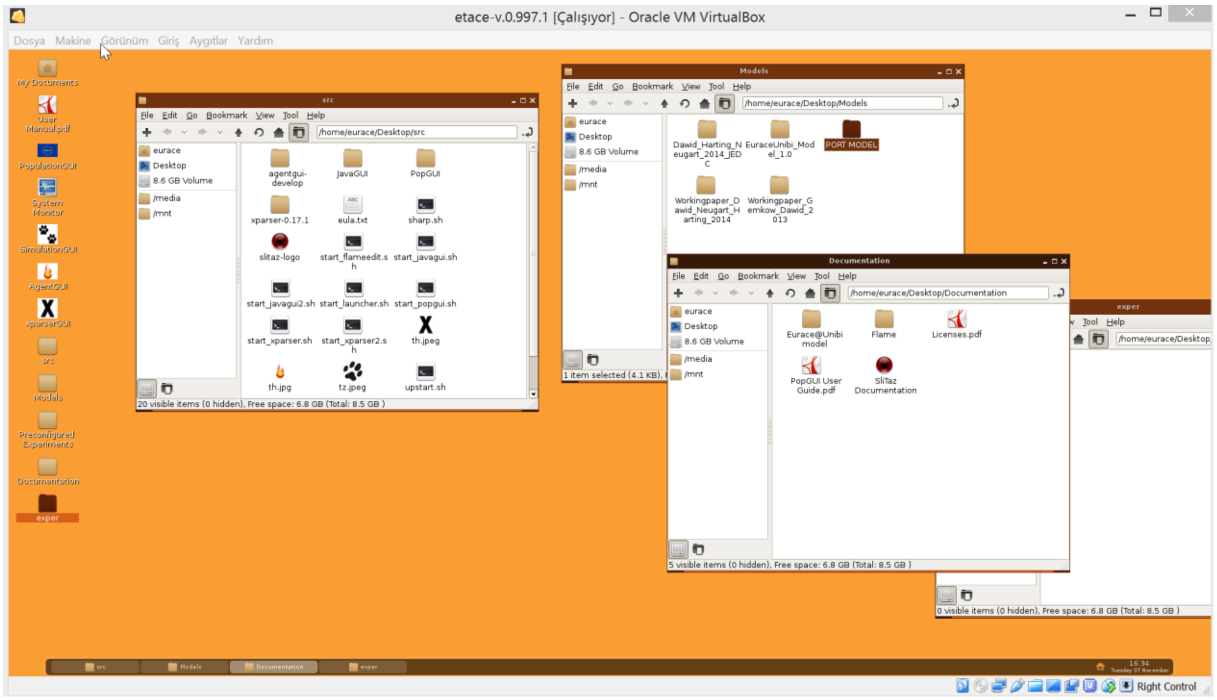

etace-v.0.997.1 [Çalışıyor] - Oracle VM VirtualBox
Dosya Makine Görünüm Giriş Aygıtlar Yardım
src
Models
Documentation
exper
/home/eurace/Desktop/src
/home/eurace/Desktop/Models
/home/eurace/Desktop/Documentation
20 visible items (0 hidden), Free space: 6.8 GB (Total: 8.5 GB)
5 visible items (0 hidden), Free space: 6.8 GB (Total: 8.5 GB)
0 visible items (0 hidden), Free space: 6.8 GB (Total: 8.5 GB)
Right Control

Figure 30 ETACE-Virtual Machine

In Figure 30, the interface of the Virtual Machine can be seen. The working principle of the VM is the same with the model that is run on the Windows. But it is worth saying that it is faster than it. But usage of VM can cause some problems. Even if there is no problem on Xparser as code or model. But it is better to know that VMs are suitable for HPC (High Performance Computers).

It is a tool that helps to analyze the model with using programs available on the Virtual Appliance sand its respective tools with respect to the FLAME. Virtual appliance, is used, has been created at ETACE [86]. The main idea of this software is to collect data as easy as possible related to the initialization, execution, modification and finally analysis of the agent-based model [81]. It is totally free software and another important point, the software is totally open source and open access also. It is developed with minimum performance configuration. If developer wants to increase it, the number of processors and allocated memory size have to be increased.

The software offer us a desktop after the installation with the folders. These are Documentation, src, Models, exper and PreconfiguredExperiments. As a briefly, Documentation folder includes the licences, manuals etc. SRC folder includes the source files about the applications. Models file has executable version of the model. Exper folder is empty but it helps to collect the experiments of the new simulations of the model. Preconfigured Experiments folder contains ready to run simulation setups.

The VA launches GUIs also. PopulationGUI for the initial state as input of the simulation, AgentGUI for the model structure, xparserGUI for parsing and compiling the model and SimulationGUI for starting simulations.

2.3.6. Development of the code and debugging procedures

It is worth saying that the variables, the functions and the messages are decided and it is time to mention about the implementation process. Before starting the procedure that was developed as previously expressed and the cyclical check and its correctness.

To do this, debugging operation during development, data printing functions are added to the code after each variation, active when the global constant debug_mode is set to 1 during the use of the PopGUI program. These printing functions then presented us with all the data of the program each time they were modified, i.e. after each function, so that the data can be checked and understood where any errors occurred.

Another consideration, all variables will be random at the end of the project, in particular all times (those called time_name, and consequently also the name_time_end) and the ENS, as deterministic variables, assigning them a fixed value included in their range of variability. Specifically, the following were set as ENS = 1, MMA_time = 5, DEL_ORD_time = 15, CUSTOMDECLARATION_time = 2, OT_time = 5, GATEIN_time = 120, NO_time = 5, PICKUP_time = 5.

This was done because in the development phase it is extremely complicated to use a code comprising stochastic elements, since the randomness that they bring does not make it possible to fully understand the functioning of the system.

So, every time the positive feedback was gotten from the program, and modifying these variables are started to make them random. Their insertion was done gradually, observing as the program did not have problems. In case these were found, the code was acted again carefully, and using the results helped us (both the txt and the database) to understand where the error could reside and that section are modified.

In the final version of the program the ranges of the variables is become the following:

• ENS variable between 0 and 1 with equal probability;

• MMA_time variable between 2 and 40;

• DEL_ORD_time variable between 15 and 240;

• CUSTOMDECLARATION_time variable between 1 and 40;

• OT_time variable between 5 and 60;

• GATEIN_time variable between 120 and 480;

• NO_time variable between 5 and 240;

• PICKUP_time variable between 5 and 60.

These are the values that are used by the various documents to be managed or produced and were given to us as project specifications.

The problems that had to be managed related to the variables that can be entered by user ENS_MAX and CUSTOM_WORK_MAX: given their significance it can be understood that the developer/user can find itself in the presence of a crowding of containers not processed within the customs if the second is less than the first (a bottleneck). Precisely for this reason, the arrays are added to the model that contains the number of ENS and the number of arrival ID at Customs, namely ENSC_LIST and ENSC_ARRIVALID_LIST.

This management of the processed containers has not been modified several times, finally decides for the FIFO (First In - First Out, which is the first arrived, is processed first).

As an example, in the Appendix section, the code for the "AgentMaritime_send_ens" function is shown.

Then, model execution process is started. FLAME is used that contains xparser to parse a model XMML into simulation program source code that can be complied together with the agent functions implementation program source code.

As it is described on previous chapters, xparser generates simulation source code files in the same directory as the model file. The files such as stategrapgh.dot, latex.dot, makefile, xml.c etc.

The simulation source code files then require compilation, which can be easily achieved using the included compilation script ‘Makefile’ using the ‘make’ build automation tool. The program ‘make’ invokes the ‘gcc’ C compiler, which are both free and available on various operating systems.

When a simulation is running after every iteration, a states file is produced in the same directory and in the same format as the start states file with the values of each agent’s memory.

Each agent has a memory that holds variables. Transition functions can read and write to variables in the agent’s memory. Communication between agents is achieved via messages. Transition functions can also read incoming messages and write outgoing messages.

2.3.7. The Adjustments of Simulation Processes

The simulation range is decided firstly. Three different quantities of the iterations are settled as 500, 1000 and 1500. All these quantities are placed on the launch.bat file of the main model. It can be seen in Figure 28.

There were two variables that have to be changed during the simulation for every iterations. These variables help us to get different results in different situations. These are ENS_MAX and WORK_MAX.

ENS_MAX → The first communication will occur between the Agent Maritime and the Customs: the first thing that AM will send a note with an ENS document to inform Customs about incoming goods (Entry Summary Declaration, is a conditional document that depends on the type of travel that the goods perform: when presents to the customs, if the goods come from outside of CEE(Central or eastern Europe) countries, the ENS is obligation) As it can be understood that for each incoming container the AM will prepare one ENS.

int ens (on the flame model); ENS document for every container (if it is 1 means there is container, if it is 0 means there is no)

int ens_max; Maximum number of containers that the AM can process at the same time (is selected by the user in PopGUI).

ENS_MAX is chosen always 10 in the simulations.

int work_max; The maximum number of containers that Customs can work at the same time (is selected by the user in PopGUI)

WORK_MAX is chosen as 1, 2, 4, 6, 8, 10 in the simulations. Table 10 shows us the iterations that variables are changed.

Simulation ID	Iterations	ENS_MAX	WORK_MAX
1	500	10	1
2			2
3			4
4			6
5			8
6			10
7	1000		1
8			2
9			4
10			6
11			8
12			10
13	1500		1
14			2
15			4
16			6
17			8
18			10

Table 10 Changed Variables on iterations

Briefly, the parameters that are modified during the simulations, the times that came in the Project specifications, the possibilities of creation of certain events such as arrival of the container, acceptance of the documents, the maximum workload of agents, for example two variables ens_max and work_max that can be changed(as it is said before) with using POPGUI. These are indicated respectively as the max number of containers that the Agent Maritime can process at the same time and the max number of container that Customs can work at the same time. Another important point, it is mentioned above is the number of the iterations. It means how long the simulation will be.

Chapter 3

Literature of the Port

3.1. The Port

A port is the place where the goods may be shipped for export or import. In addition, the ports are correlated with maritime trade and carrying cargo. Railways, automobiles, trucks, and also airplanes can be counted as a types of transportation and they also use ports as seaport or airports. On the other hand terminal is a part of the port that is specialized to keep the goods inside such as containers, cars, woods, people etc [43].

Furthermore, the port is a structure that is naturally or artificially located on the maritime coast which is able to allow to berth and to moor of the boats and ships. Also it provides protection from adverse conditions. Another important role of the port is to allow and facilitate the exchange of the goods and boarding of people.

There are different types of port. These are the commercial one which is used for the transfer of the goods and the embarking of people, the tourist one or the one that is dedicated to the only recreational purposes and the last one is the military one.

The physical structures of the port are similar but only the commercial ports will be dealt as given in the nature of this research.

The commercial port is a very complex environment.

It is made up of various elements differ from other types [90]: First, external piers. These are protected area surrounds by concrete blocks or large breakwaters on the side towards the sea. Internal piers and docks that helps to dock ships easily and allow them to get on and off or load and unload. The headlights, to show visibility of the port. Moreover, cranes, to load and unload goods on ships, to put them to the sea and to move goods inside the port. Warehouses or yards to store goods or containers inside the port, Terminal for goods and passengers (usually also there are terminals for the boarding of the passengers). The roads and also the railways to reach it for the goods.

The port has a set of many different actors that coexist and work at the same time to achieve same purpose. This purpose is to promote the trade different types of goods by the ships and improve it over the time.

All commercial ports operate roughly the same and have approximately the same entities to work.

Port operators contribute to all port processes to complete them from transport of goods to the destination and vice versa. In Italy port operators contribute to completing a port process, that is the transport of goods to the destination and vice versa. A part of them belongs to the "maritime" and they have titles and qualifications to perform various tasks that special profile of safety of navigation are required. They are the maritime workers (the master, the officers, the boat, the cook, the hub and the small kitchen on board the ships that offer bunkering services, clean water, removal of waste from ships and so on), the pilots of the harbor, boatmen, personnel of Coast Guard. In addition, another part of the workers belong to the Public Administration (also called Port Authority) and they are not different from other officers with the exception of specific skills in the port. These are the workers of the Port Authority, Customs Agency, Guardia di Finanza (Finance Police), Maritime and Veterinary Health Service, Maritime Border Police, Firemen and territorial control agencies (such as 'ARPAL in Liguria) etc. Another category of the works are port employees, i.e. port operators such as administrators, companies, IT technicians, personnel assigned to the management, selection and training personnel, workers (can be on the ship, on construction sites or in the gate). Experts can perform tasks related to planning, scheduling, control, coordination, execution and maintenance. Finally, the works that belongs to the category of transport agencies that are represent the ship which is in transit and being in charged with administrative point of the government, responsible of the operation for the terminal of arrivals and also the commercial carriers and the shippers and receivers. They also work as mediators between ship owners and freight shippers for the transportation and ship sales.

The shipping agents represent the goods in transit and observe them during the journey from the sender to the buyer, managing relations with shipping companies, terminal operators, land transport and customs.

The process of customs clearance of goods entered to the port, is simulated in this project, and requires several steps and a lot of documentation produced by the agents above but this will be described later during the exposure of the procedure that is followed to build the simulator.

As seen the information above, as these individuals, operators, their functions and their contributions and communications make the port very complicated and unpredictable.

Furthermore, because of this multitude variety of goods, agents and processes, the last ones are affected by the delays that is called ‘bottlenecks’.

Because of this multitude of goods, agents and processes, the transporting process are often affected by delays due to the so-called "bottlenecks" [48] that often occur in certain situations: for example delays on transport, i.e. ship delays, which also generate additional costs, or those that arise on the terminals, such as the request for extra labor, the congestion of the shipyard or need for re-handling of goods. But they can still occur because of the transport companies, which generate waiting times and follow loss of business, or because of the shippers, which sometimes take longer delivery times.

The problem can be solved with increasing capacity. Increasing capacity in the terminal and also increasing as physical expansion. It means improving in the terminal performance can handle this problem with developing and implementing new IT systems, new or extra equipment, training labour force. Moreover, capacity expansion in the facilities. But the second solution can be trouble because the cities do not have enough space for them. Also investment for the new infrastructure is hard because of finding funds.

It is worth remarking that some special cases like transportation for liquids etc. are not considered because of safety reasons. The containers are reusable and standardized with ISO standards that allows us not to take into consideration about their contents during the transportation. This helps us to simplify to work related with their customs clearance. So, in the explanation of the process, containers are not mentioned but the documents related with this process are considered, only the content of the containers are excluded.

The containerization process needs too many manpower for handling and docker. In these ages, it is easier than previous years. Mechanization and computerization of the port operations has caused to drop in employment significantly. Massive cranes and machinery allow rapid and safe handling and this also helps to cut down in time and costs [91].

The structure and characteristics of a container terminal can be identified four main subdivisions as follows;

The first area is Entrance of the Terminal (also called Gate) where both in and out processes of the goods, checking and registration processes of the containers, administrative and other customs procedures are being carried out. There are finance police, customs office, control tower and other administrative offices.

The second one is the movement area for the wagons or trucks that receive and also deliver containers. It is formed as parking area for trailers. It includes lines and railways that allow the circulation of the rail in and out and the roads allow to reach warehouses where the containers are positioned for filling or emptying. The massive working areas helps to the vehicles to enter and exit from the gates easily.

The third zone is the storage. It is a kind of stack where containers are transported to and also placed. There are also particular areas that are designed for dangerous goods and reefers.

Finally, the fourth zone is ship-to shore. It is worth noting that it is operating area too. Containers are moved from ship to berth. Quay cranes help to a ship for loading and unloading containers. These cranes have high operating speeds for lifting and moving. There are also cranes that has platforms helps to move containers to the rails or to the wheels.

The structure of the containers allows to be stacked up to four units or more in the storage areas. There are special hooks and lift that can carry and move them easily to the vehicles.

It is worth mentioning that all equipment are designed to move the boxes up and down from the ships and place them in storage area where they wait to be shipped or collected. It is perfectly fitted that there is no wasted place and ships can stand side by side and containers can be stores also there where the stores are waterproof so do not need to be in covered area such as warehouses in quayside. It is worth noting that areas that are assigned for the customs control are covered. Then the stacked containers are forwarded to two different zones. The first zone is dedicated to containers that wait for exporting and then they will be loaded to the ships. The second zone is dedicated for importing. The goods are going to be loaded for sending to various destinations.

In a modern container terminal will have giant portal crane or transtainer which is able to reach the top for loading or unloading goods to the ships. This portal cranes can move vertically and also horizontally when it is stationary.

Planning is the most important feature of sorting operations. Because the designers of the terminal want to reach max level of speed in handling containers and minimum number of operations. For this purpose, the key is to know where the container is and deliver them in the exact sequence and avoid double movements during unloading.

The container terminals have progressively limited human intervention in the port operations because of the operational necessity and cost reductions. IT development for driving mechanicals equipment, automation for lifting and handling, sensors and guided roads help to carry autonomously with greater precision, less time and security. Sophisticated computers and software have become indispensable tools and thanks to all of these, 1500-2000 containers can move per day.

3.2. The Port Of Genova

The port of Genoa is one of the largest Italian port after the extension [91], 700 hm2 of, 500 hm² of water jump and 22 km of docks and fencing at the quay, eight meters for the passenger drop and fifteen meters of the large container terminal of the VTE (Voltri Terminal Europa) and the SECH (Southern European Container Hub). It has the biggest number of the shipping lines in Italy and it is in the first place to handle containers (as final destination) generally for different volume of goods. 10 thousand direct workers and 30 thousand auxiliary staffs work in the port.

In 2015, 51.3 million tons of goods has moved. 2.2 million TEU (twenty-foot equivalent unit, approximately 6 meters that ISO standard measurement of volume in the transport and corresponds to 40 total cubic meters) containers have handled. Thanks to these numbers, The Port of Genova has become a reference point for national and Mediterranean terminal of the Rhine-Alpine - European Corridor.

Genova Port is active around € 10.9 billion of production, € 4.6 billion of added value and employs 54 thousand units of work, weighing approximately 10.8% of the added value of Liguria and 8.3% of employment only in Liguria.

Moreover, the port produce 3.2 billion euros (equal to 12,6%) and 37 thousand working units (9.7% of employment) only for Metropolitan City of Genova.

It is worth taking throughout Italy, values are double. It contributes more than 9.5 billion euros of added value and creates 122 thousand units of work.

Genoa Port is a port multi-service having 29 terminal specialized and capable to accommodate every type of ship and manage any type of cargo and passenger.

Firstly, every terminal has its own product specialization between cargo and passengers, then respect to the form of goods (dry and liquid, dissolved or contained in the holds of the ship or cargo, or special goods, perishable or non-aggregated pallets or containers) and type of passengers (ferry or cruise).

Depends on the specialization a different type of ship and terminal and port cycle is adopted. Based on the space and resources (dock, parking areas, loading equipment, work organization, safety and security, customs, transport and forwarding of arrival of goods) the cycle can change.

The port operators are the terminal operators that have a concession by the Port to manage, with their own organization and personnel, the trade of goods of the ships that embark and disembark at some port docks.

Among the operators there are also those businesses that provide services to particular port terminal operators, not having even a concession of docks (integration of the cycle, cargo handling, warehousing, consolidation).

The port operators, that contribute to complete a port process, the ship conveyance of the goods to the destination and vice versa, are a great variety.

A portion of them belonging to the "seafarer", having titles and qualifications to achieve several tasks for which special safety profiles of navigation are necessary. They are: maritime workers (the master, officers, the boatswain, the cook, the hub and the small kitchen that are on board of the ships which offer bunkering services, clean water, removal of waste from ships and so on); the pilots of the port; the boatmen; the staff of the Harbour-Coast Guard;

Another portion of workers belong to the Public Administration (Port Authority also, of course), are no different from other leaders, officials and public employees, except for specific skills in port. They are: workers of Autorità portuale, Customs, Finance Police, Maritime health and Veterinary, Maritime Border Police, Fireman, ARPAL, ASL.

Another category of workers is that of the "port employees", that is the port operators: directors, business, information technology, management, selection and training of personnel, the operator (on the ship, the side yard, side gate); in operating the expertise can be related to the planning, scheduling, control, coordination, execution and maintenance.

A portion of workers belong to the category of shipping agencies which represent the ship in transit and curate the interest from an administrative point against the government, from the operational point against the arrival terminal of the ship and also against the commercial carriers and the shippers and receivers. They also work as mediators between ship owners and the shippers of freight with regard to the transportation and sale of ships.

The forwarding agents represent the goods in transit and see them during the journey from sender to receiver and manage port relationship with shipping companies, terminal operators, the ground transportation, Customs.

Chapter 4

The Model

4.1. The Process of Import

As it is mentioned before, in general the development of a typical agent-based model consists in four steps. First, the identification of some real world streamlined phenomena of interest that the modeler wants to explain. Second, the model is described by means of the time-line of the events, the micro level dynamic equations which include the individual agents' behavior, the set of parameters, and the set of random disturbances. Third, the models output is compared with the observations gathered from real world datasets. Fourth, the model is used to perform scenario analysis by modifying some of the behavioral equations or some of the parameters.

In this section, the model description of import processes of the port is presented. It is built the model using the features of the Genoa port in structural terms. It is considered the stakeholders of the Genoa port and their mutual interactions, using the real network and the real data related to the processing of the documents exchanged among various actors (agents) included in the model.

The information about actual Genoa port stakeholders and effective dynamics has been provided by the main companies involved in the port activities within the GESTEC project [38].

It is a complex process, where a large number of agents exchange documents and authorizations through well-defined regulations, to allow goods in the containers to be unloaded from the ships, stored in the aprons(terminals) and be picked up by land carriers, such as trucks and trains.

The agent model construction is defined with the entities that are involved, the variables and functions are described such as definition of the documents or steps of the containers. On the other hand, the environment of the model that is necessary, it means global variables.

To construct the agent model, the entities involved were defined on one hand, describing the variables and functions, such as those that define documents or take into account the passage of containers; on the other, the environment necessary for the model, that is, global variables.

The port model includes the following 9 agent types: the Agent Maritime, the Customs, the Custom Clearance Agent, the Delivery Service, the Deliverer, the Terminal, the Finance Police, the Gate, Terminal Area.

Figure 31 shows a graphical representation of the present port model in terms of agent classes (ellipses) and documents (messages that is send by the one agent to another.) exchange flows (arrows).

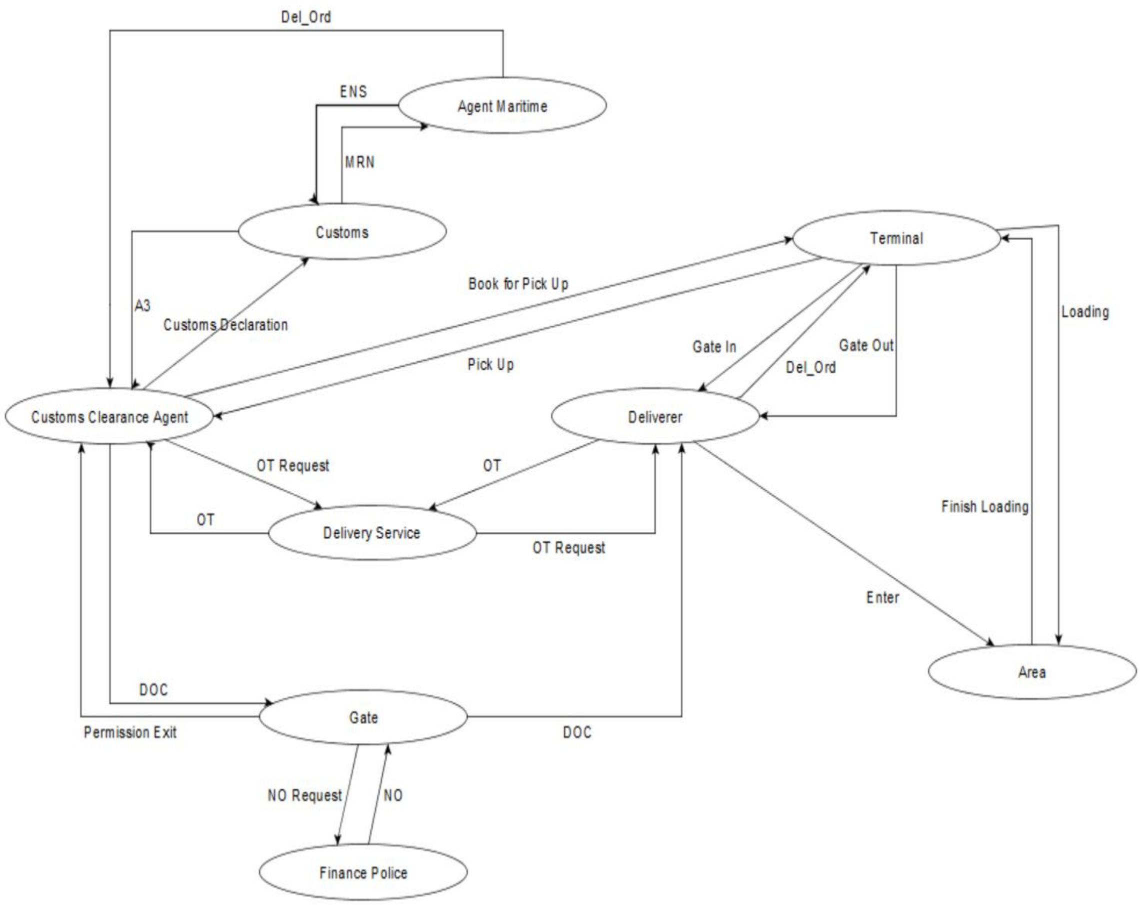


Figure 31 Port Model (Graphical)

The Names of the documents exchanged are reported in Table 10.

Symbol	Description
ENS	Entry Summary Declaration
MRN	Movement Reference Number
MMA	Manifesto Merci in Arrivo
DO	Delivery Order
CD	Customs Declaration
A3	Customs Entry
CC	Clear Code
OT	Order of Transport
SC	Security Clearance
DOC	Leaving documentation

Table 11 Documents produced during the import process

Each agent type is defined by its predetermined actions which are induced assigning specific states and sets of functional attributes, properties, or rules by means of predefined parameters. Hereafter, we describe briefly the duties performed by the port operators (agents) involved in the port model.

In another words, all agents has their own structures that is formed and developed with their own variables and functions. Also the messages that they need to communicate with each other. (This part will be explained detailed in the tools that are used during the simulation creation section.)

4.2. The Guideline of the Processes

The process of the model begins with the arrival of the ship to the port. After that, the first communication will occur between AgentMaritime and the Customs. Agent Maritime will send a note to the Customs with an ENS (Entry Summary Declaration) document to inform it about the incoming goods. ENS is a conditional document that depends on the type of the journey of the merchandised goods. If the incoming goods do not come from the EEC countries, ENS is mandatory. So Agent Maritime will have to prepare one of these documents. Furthermore, the customs will assign the codes that will be produced and assembled for generation the MRN document. Movement Reference Number is assigned to all computerized documents for export or transit. The AgentMaritime will receive the MRN document and it will send the MMA document as a response to the customs and the delivery order to the Customs Clearance Agent. Moreover, MMA is the Manifesto Merci in Arrivo. There are indicated various items to identify each container that is boarded on the ship. There are the disembarkation place at the port where the terminal served. Most significant notes on the manifesto, the name of the ship, the port of arrivals, the arrival time and other general information. Other one is delivery order. It is an authorization to the deliver for registration of the goods.

Subsequently, the customs will send the document A3. (It is Customs Entry that consists of: number, customs section, date, weight and package, it is a component of the MMA) to the customs clearance agent who will send the book for pick up to the terminal and then the customs declaration to customs. Custom Declaration is the document which declares the value of the goods that is passing through too customs.

The Custom Clearance Agent waits to receive three documents from other agents. These are; firstly, the Clear Code from the customs. It is the authorization to pass. The second is the Pick Up from the terminal. It is authorization to collect container if there is payment for service performed and as the last, Delivery Order from the Agent Maritime.

The customs clearance agent will wait to receive 3 documents: the "Clear Code" from the customs (i.e. the authorization to pass the latter), the delivery voucher from the terminal (i.e. the authorization to withdraw the container only if there is the payment of the services performed) and the order of delivery by the Maritime Agent. At that point, CCA will send transport order to the delivery service. And then it will forward it to the Deliverer. Deliverer will confirm it and send the OT (Transport Order) document back to the Delivery Service.

The Delivery Service will forward the transport order to Customs Clearance Agent. The communication between Custom Clearance Agent and Gate will start. First the CCA will send the exit document to the gate and if the container receives the authorization to exit. The goods will be released.

After the goods are released, exit document will be send to the Deliverer that includes date to pick up of the goods. The clearance of the goods is necessary and it will be produced by the Finance Police to be able to collect the goods. The container must be checked to prevent that contains dangerous, counterfeited or illegal goods.

Then the deliverer will send the Delivery Order to the terminal where it will request access. If the terminal agree, it means if the documents are correct and there are no legal issues, the deliverer will be authorized to enter the terminal.

Now, it is necessary to search and load the container. For loading container, the process have to wait until the Area Agent and cranes are free. Then truck or train will be used for load as well as searching for container will be ended.

At this point, the Deliverer receives permission to leave the terminal.

If the processes and the variables is known and wanted to performed, the development of the agent model will begin and it is described on the part of Agents' implementation.

4.3. Agents of The Model

The agents that have been developed are 9 listed in alphabetical order;

- *AgentMaritime*, maritime agent.
- *Area*, where the containers stored.
- *CustomClearanceAgent.*
- *Customs.*
- *Deliverer,* carrier.
- *DeliveryService,* forwarder.
- *FinancePolice,* control authority.
- *Gate.*
- *Terminal.*

A. Agent Maritime (AM)

Agent Maritime is the actor responsible of the incoming goods. He follows the declaration process of the goods/products. He interacts with the customs by means of the Entry Summary Declaration (ENS) and the “Manifesto merci in arrivo” (MMA) documents, and with the custom clearing agent by means of the Delivery Order (DO). The ENS is a document that informs the Customs about the incoming goods, whereas the MMA is a document with the complete list of goods. In particular, MMA identifies each container embarked in the ship and includes information about the name of the ship, the port of arrival, the time of arrival. DO is a document necessary for entering in the port and picking up the goods.

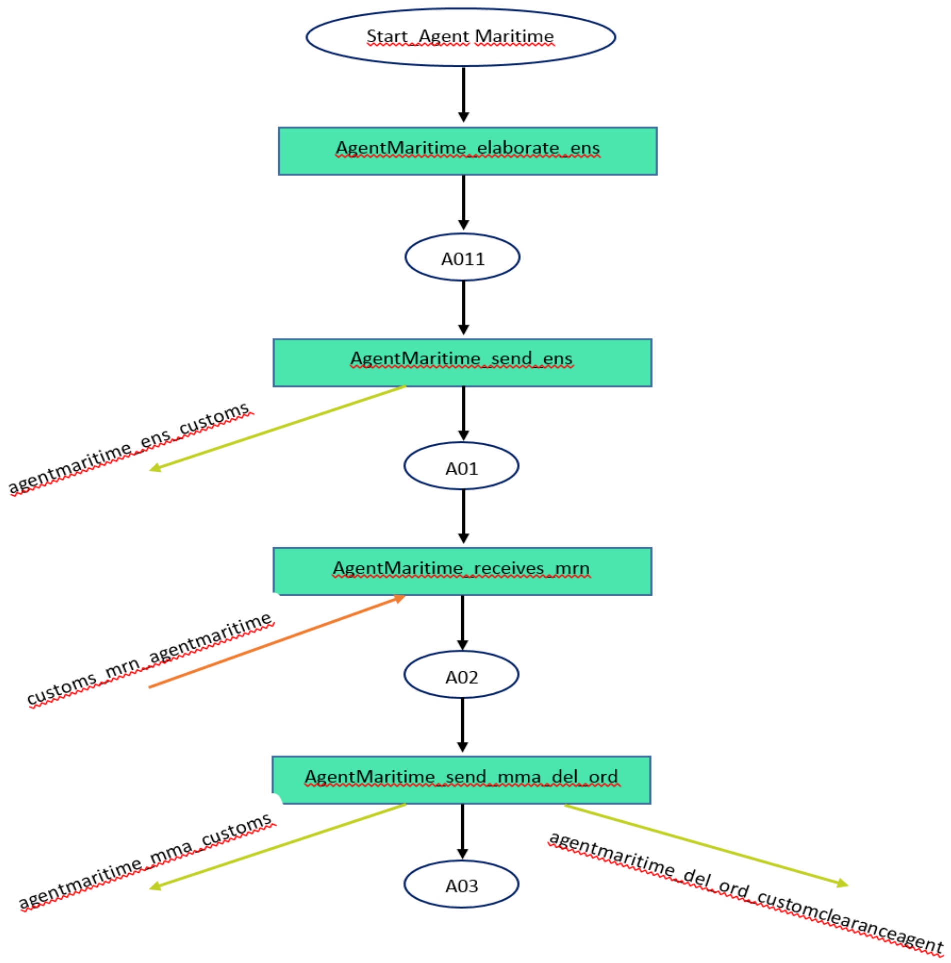

Start_Agent Maritime
AgentMaritime_elaborate_ens
A011
AgentMaritime_send_ens
agentmaritime_ens_customs
A01
AgentMaritime_receives_mrn
customs_mrn_agentmaritime
A02
AgentMaritime_send_mma_del_ord
agentmaritime_mma_customs
agentmaritime_del_ord_customclearanceagent
A03

Figure 32 Agent Maritime Stategraph

The first communication exchange happens between Agent Maritime and Customs, that is the Agent Maritime sends the Customs a notice using an ENS document to inform it about the incoming goods. The Customs assigns codes which are assembled to generate the MRN document. The Agent Maritme receives the MRN document and in turn sends the Customs and the Custom Clearance Agent the MMA document and the Delivery Order respectively.

B. Customs (C)

Customs checks if the imported products are conformed with the European and/or National laws. Operatively, Customs assigns an identification number to all the imported goods, that are collected in the Movement Reference Number (MRN) document. Moreover, the MRN contains information for export or transit. Customs interacts with the Customs Clearance Agent by means of the Customs Entry (A3) and the Clear Code (CC) document. The A3 document contains all the relevant information related to the goods such as date, weight and packages, whereas CC contains the allowed clearance.

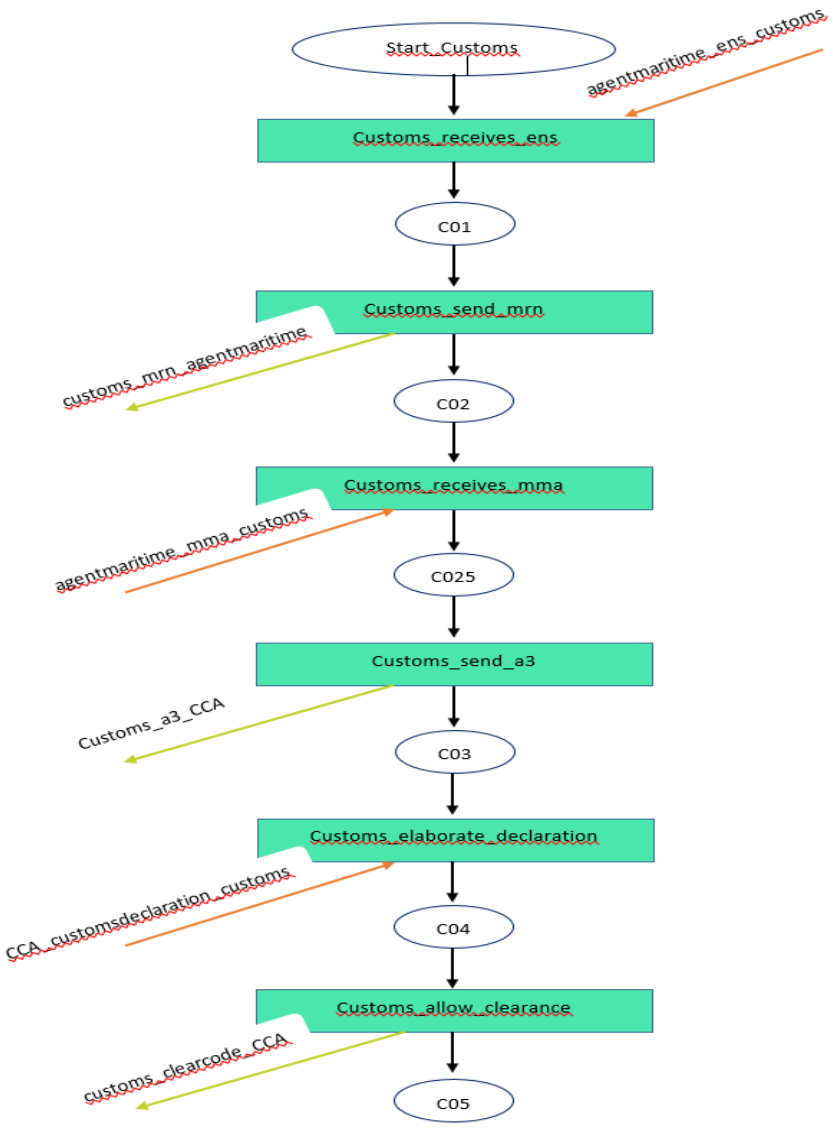


Figure 33 Customs Agent Stategraph

Customs sends the A3 document to the Custom Clearance Agent who sends the Terminal and the Customs the booking for pick up and the Customs Declaration respectively.

C. Customs Clearance Agent (CCA)

Customs Clearance Agent (CCA) is responsible for preparing the documentation and arranging the pick of the imported goods. CCA interacts with the Customs by means of the A3, the Customs Declaration (CD) and the CC documents. CD is a form, required by C, compiled by CCA that declares the value of the imported goods. Moreover, CCA interacts with the Delivery Service to organized the pick up of goods and with Gate for the leaving process by means of the leaving documentation, i.e. all the relevant documents necessary to leave the terminal area.

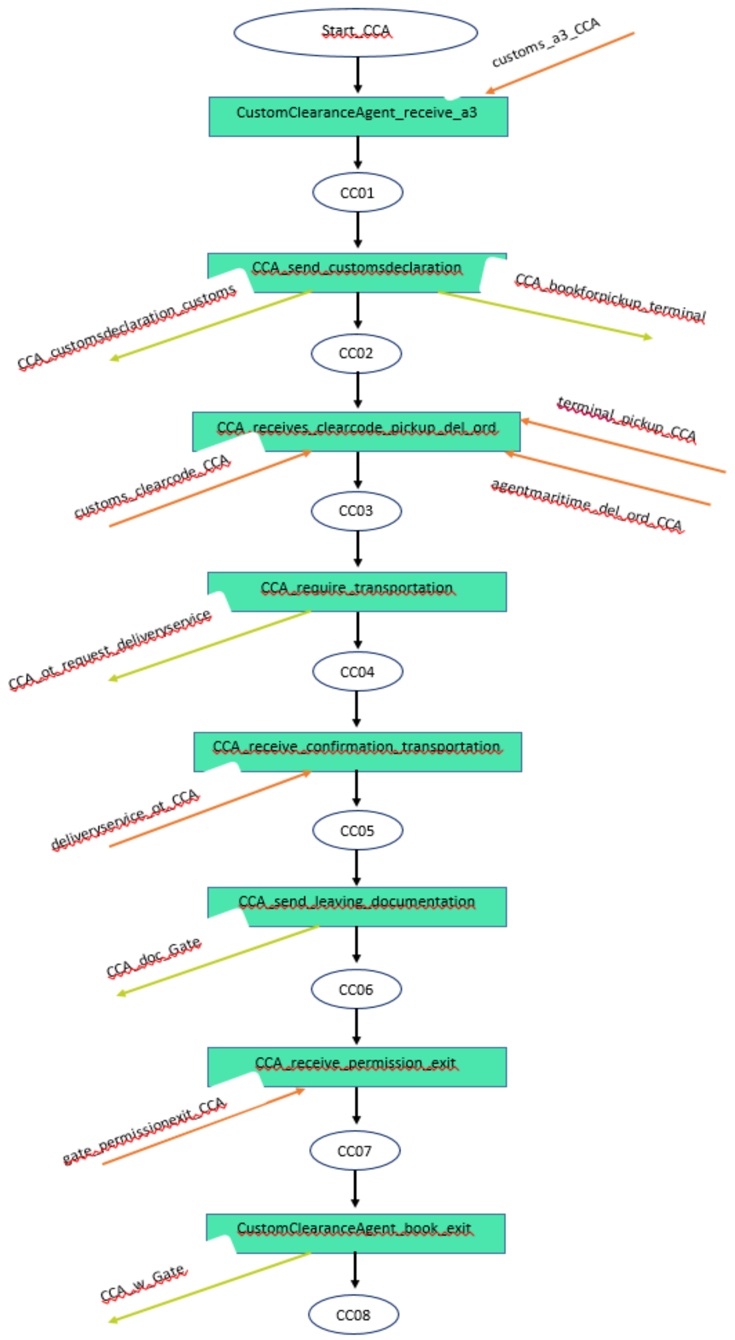
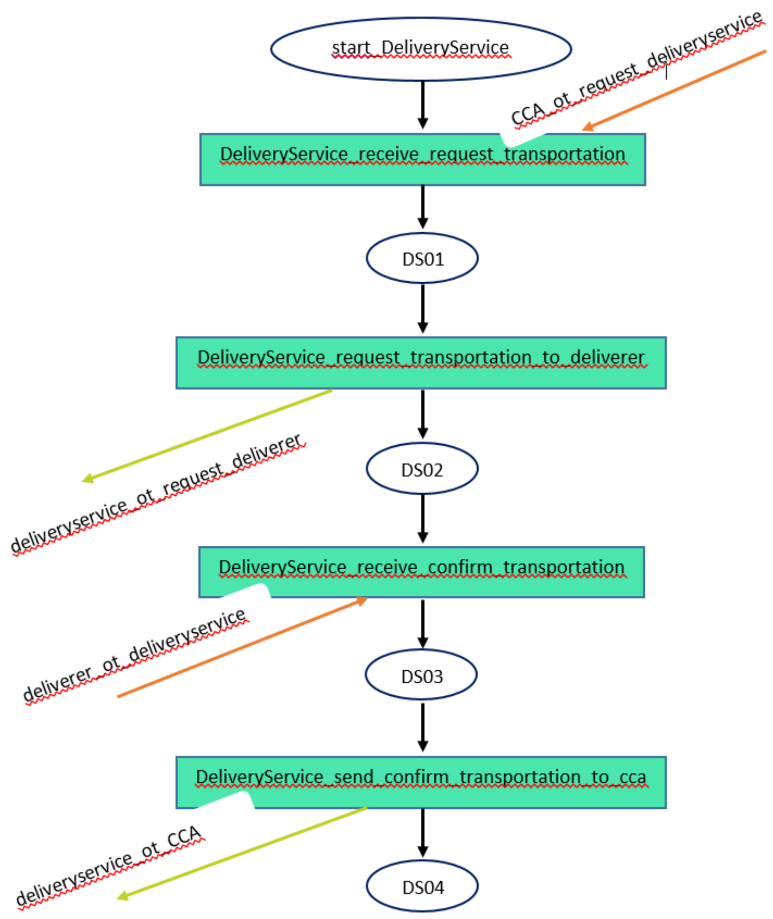

Start_CCA
customs_a3_CCA
CustomClearanceAgent_receive_a3
CC01
CCA_send_customsdeclaration
CCA_customsdeclaration_customs
CCA_bookforpickup_terminal
CC02
CCA_receives_clearcode_pickup_del_ord
terminal_pickup_CCA
customs_clearcode_CCA
agentmaritime_del_ord_CCA
CC03
CCA_require_transportation
CCA_ot_request_deliveryservice
CC04
CCA_receive_confirmation_transportation
deliveryservice_ot_CCA
CC05
CCA_send_leaving_documentation
CCA_doc_Gate
CC06
CCA_receive_permission_exit
gate_permissionexit_CCA
CC07
CustomClearanceAgent_book_exit
CCA_w_Gate
CC08

Figure 34 Customs Clearance Agent Stategraph

The Custom Clearance Agent having received the Clear Code from the Customs, the permission to pick up from the Terminal and the delivery Order from Agent Maritime requires transportation to the Delivery Service who sends the Deliverer the transportation request.

D. Delivery Service (DS)

Delivery Service agent (DS) is responsible for managing the transportation requests. DS interacts with the CCA receiving the request of transport and with the deliverer agent arranging the transportation by means of the Order of Transport (OT). OT is a document that indicates when it is possible to pickup the goods from the terminal area. It is also required in the picking up process.



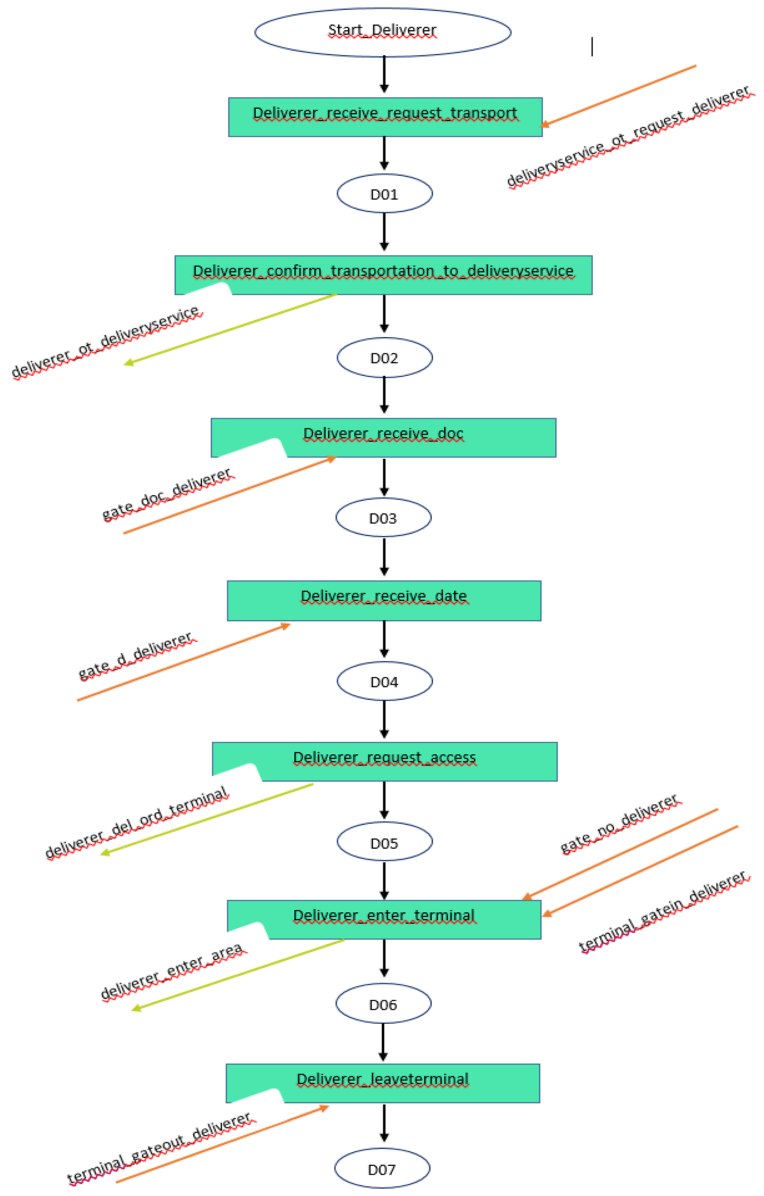
start_DeliveryService
CCA_ot_request_deliveryservice
DeliveryService_receive_request_transportation
DS01
DeliveryService_request_transportation_to_deliverer
deliveryservice_ot_request_deliverer
DS02
DeliveryService_receive_confirm_transportation
deliverer_ot_deliveryservice
DS03
DeliveryService_send_confirm_transportation_to_cca
deliveryservice_ot_CCA
DS04

Figure 35 Delivery Service Stategraph

Deliverer confirms to the Delivery Service the transportation using the Order of Transportation document then the Delivery Service gives the Custom Clearance Agent the Order of Transportation and the communication between the Custom Clearance Agent.

E. Deliverer (D)

Deliverer Agent interacts with DS compiling the OT document. Moreover, D interacts with Gate agent by means of all the leaving documentation. Finally, D interacts with the terminal agents to get the permission to enter the terminal and with the terminal area agent to arrange the pick up.



Start_Deliverer
Deliverer_receive_request_transport
deliveryservice_ot_request_deliverer
D01
Deliverer_confirm_transportation_to_deliveryservice
deliverer_ot_deliveryservice
D02
Deliverer_receive_doc
gate_doc_deliverer
D03
Deliverer_receive_date
gate_d_deliverer
D04
Deliverer_request_access
deliverer_del_ord_terminal
D05
gate_no_deliverer
terminal_gatein_deliverer
Deliverer_enter_terminal
deliverer_enter_area
D06
Deliverer_leaveterminal
terminal_gateout_deliverer
D07

Figure 36 Deliverer Stategraph

F. Terminal (T)

Terminal interacts with the CCA for allowing or not the pick up of the goods, according to the presented documents, i.e., Delivery Order and Security Clearance. If they are correct and there are no legal problems in the presented documentation the authorization is given. Moreover, T interacts with D and the terminal area agent to organize the picking up.

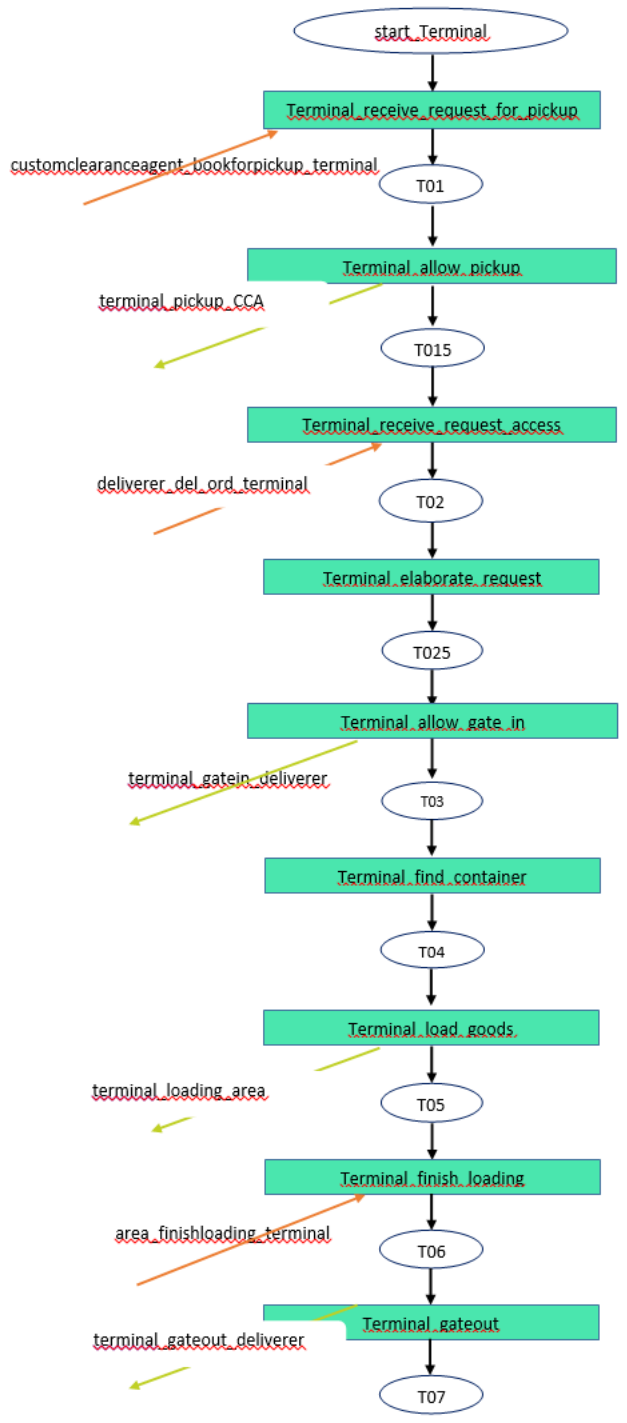

start_Terminal
Terminal_receive_request_for_pickup
customclearanceagent_bookforpickup_terminal
T01
Terminal_allow_pickup
terminal_pickup_CCA
T015
Terminal_receive_request_access
deliverer_del_ord_terminal
T02
Terminal_elaborate_request
T025
Terminal_allow_gate_in
terminal_gatein_deliverer
T03
Terminal_find_container
T04
Terminal_load_goods
terminal_loading_area
T05
Terminal_finish_loading
area_finishloading_terminal
T06
Terminal_gateout
terminal_gateout_deliverer
T07

Figure 37 Terminal Agent Stategraph

G. Gate (G)

Gate agent is responsible for exit permission of the goods. G sends D the exit permission documents, i.e., the leaving documentation, received by CCA, with the date for getting the goods and SC document, received by FP.

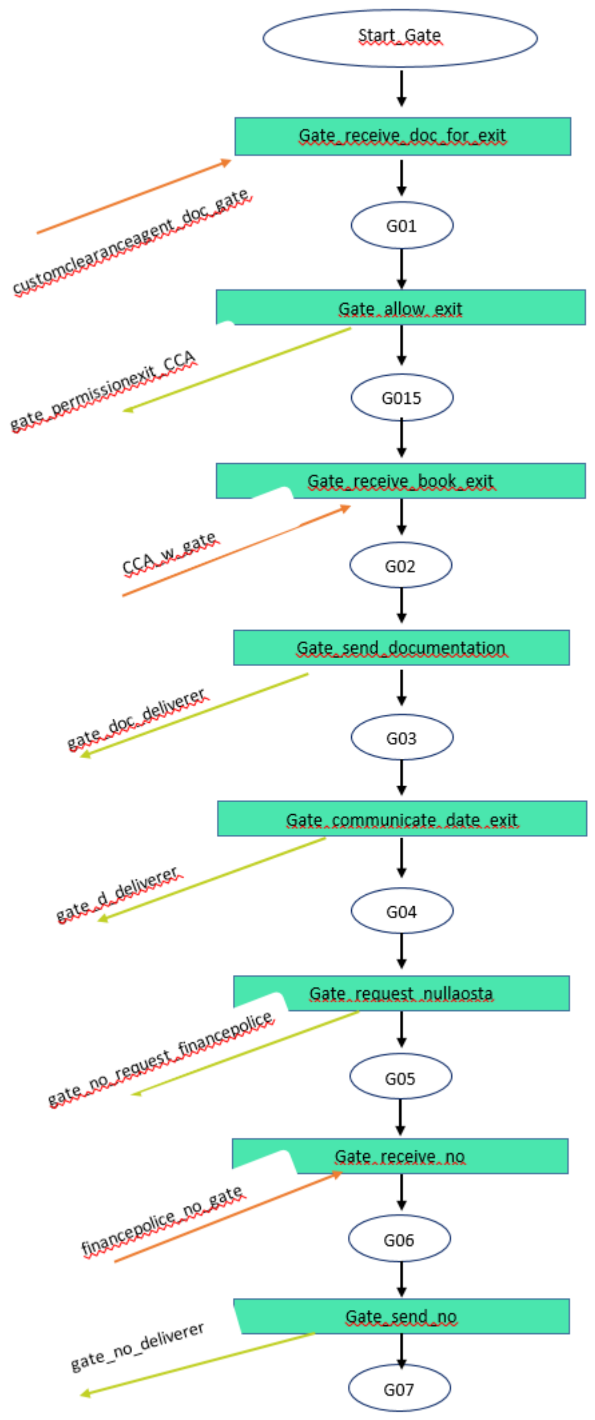


Figure 38 Gate Agent Stategraph

Then, Gate begins and the Custom Clearance Agent sends the Gate the leaving documentation and, if receives the permission to exit, books the exit.

H. Finance Police (FP)

Finance Police Agent has the task of checking the containers and preventing smuggling or trade of illegal, counterfeit or dangerous goods. In order to authorize the purchase or the pick up of the goods, the Finance Police Agent grants the Security Clearance (SC) document and sends it the Gate Agent. The Security Clearance is a document produced by the Fiance Police that guarantees that everything is legal.

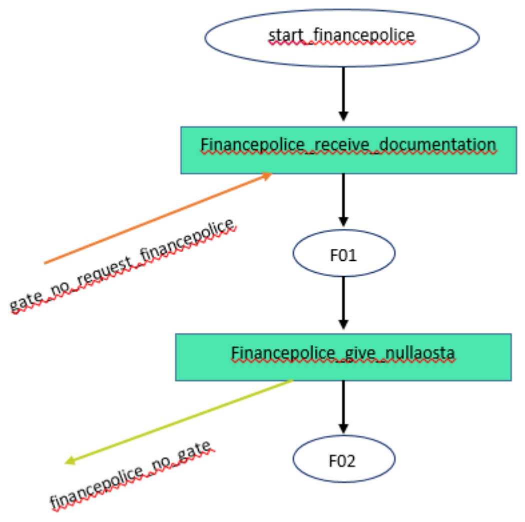

start_financepolice
Financepolice_receive_documentation
gate_no_request_financepolice
F01
Financepolice_give_nullaosta
financepolice_no_gate
F02

Figure 39 Finance Police Stategraph

Gate sends the deliverer the leaving documentation and the date to pick up and requires the Finance Police the Security Clearance which is given the Gate.

Then the Deliverer sends the Terminal the Delivery Order to require the access and, if the Terminal allows it, the Deliverer, having the Security Clearance enters the Terminal. Into the Terminal the operations of search for goods and loading of them are happen, then the Deliver receives the permission to leave the Terminal.

I. Terminal Area (TA)

Terminal Area agent is the place where the goods are, waiting the pick up. TA interacts with the T confirming the loading operations.

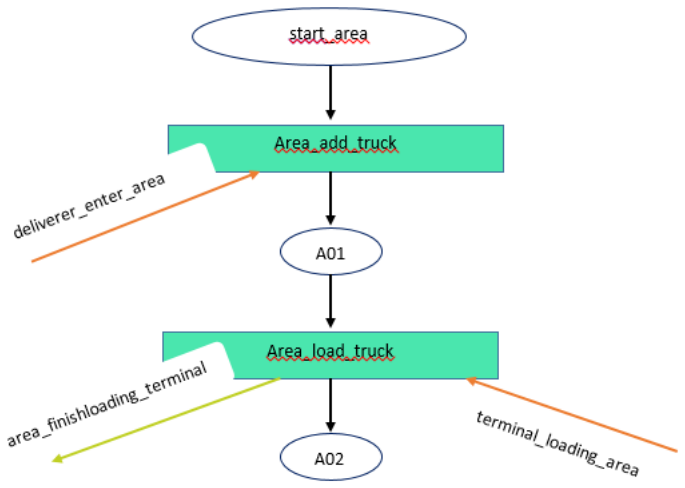


Figure 40 Terminal Area Stategraph

J. Import process

The beginning of the process is the arrival of a ship. In fact, when the ship arrives, the Agent Maritime sends the Entry Summary Declaration (ENS) to the Customs. It is worth remarking that the Entry Summary Declaration is a document that informs the Customs about the incoming goods.

The Customs, received the ENS document, assigns a code to each good and assembles all the code in the MRN document and sends it back to the Agent Maritime. The Agent Maritime, received the MRN document, sends the MMA document to the Customs and the DO document to the Custom Clearance Agent, respectively. The Customs, received the MMA, sends the A3 document to the Custom Clearance Agent, that, received the A3 document, sends the Customs the Customs Declaration (CD) and contacts the Terminal in order to arrange the book for picking up the goods. The Customs, received the compiled Declaration Form, elaborates it and if there are no errors it allows the clearance and sends back to the Custom Clearance Agent the Clear Code (CC). It is worth noting that the Clear Code is a document that allows to pick up the goods from the Terminal Area.

The Custom Clearance Agent, received the Clear Code (CC) from the Customs, the Delivery Order from the Agent Maritime and if the terminal allows the picking up of the goods, contacts the Delivery Service to arrange the transportation.

The Delivery Service, in turn, contacts the Deliverer sending the Order of Transport (OT).

The Deliverer confirms to the Delivery Service the transportation using the Order of Transport document, compiled with the delivery time.

The Delivery service gives to the Custom Clearance Agent the confirmation of the Deliverer agent sending the compiled OT.

The Customs Clearance Agent, received the Order of Transport sends all the documents to the Gate agent in order to arrange the exit process.

The Gate, received all the leaving documentation, checks that everything is correct and if so, contacts the Customs Clearance Agent to inform him that it is possible to exit from the gate. The Custom Clearance Agent, received the permission to exit, book a date to exit from the Gate agent. The Gate agent can accept of change the proposed date and if accepted, sends all the leaving documentation approved and the date to exit to the Deliverer agent. Moreover, Gate agent requires the Security Clearance (SC) document to the Finance Police. It is worth remarking that the Security Clearance is a document produced by the Finance Police that guarantees that everything is legal. If all the documentation is correct, the Finance Policy sends the Security Clearance (SC) to the Gate, that sends it to the Deliverer.

In the meantime the Deliverer requests access to the Terminal sending the delivery order. The terminal elaborate the request and if everything is right he allows the Deliverer to gate in. The Deliverer, received the gate in permission from the terminal and the Security Clearance from the gate enters in the terminal area for the loading process. The Terminal after the operations of search for goods and loading of them gives the Deliverer the permission to leave the Terminal.

All the documents produced for the import process and exchanged by the agents are summarized in Figure 31.

Chapter 5

Results and Conclusion

5.1. Objectives of the study

The ultimate goal of the thesis is to model the process of importing and clearing goods and creating an agent software framework to simulate this process.

The validation of the model was the verification of the correctness of the development of authorizations, as in the port process these have a certain order and their timing.

After implementing the software it was possible to use this tool for computational experiments, i.e. to do what-if analysis to identify bottlenecks in the process and to discriminate the process time of the individual container.

For the first point we will try to observe where these are formed and why, also trying to quantify the obstructions that are occured so the time dimensions are decided in order to determine the waiting times and their increase or decrease.

The second point was chosen because in this way it is possible to show process time of the single container, to give a measure of the waiting times related to the process.

The parameters that are modified during the simulations were the time states, which were part of the project specifications; the possibility of realization of events, such as the arrival of containers or the acceptance of documents; the maximum working quantities of the agents, ie the two variables that can be set through the PopGUI program ens_max and customs_work_max, which respectively indicate the maximum number of containers that the Maritime Agent can process simultaneously and the maximum number of containers that the Customs can work simultaneously; and the number of iterations, or how long the duration of simulation.

The two variables ens_max and customs_work_max are perhaps the major change that is made between one simulation and another. They have been varied within a low range (both from 1 to a maximum of 5), because it is similar to the real situation.

When the first one will be higher than the second one, obviously a queue will form inside the Customs, because this will only work a fixed number of containers and, if they arrive more, it will be forced to put them in a line.

In particular, the simulations were developed on a system that contains 9 different agents (the described above) all in single quantity, with a number of iterations (corresponding to minutes) variable in the range 500-1000-1500, with ens_max and customs_work_max of values 1, 2, 3, 4 or 5 independent of each other (for example the simulation can be decided with ens_max = 2 and customs_work_max = 1), because they are similar to the situation that may be encountered in the Port of Genoa.

The following chapter shows some graphic results obtained by MatLab,in random order; sizeend of the various agents, dimension of the queue and container in the queue in Customs Agent, time diagrams that represent the instant entry and exit of the containers in the Deliverer Agent.

5.2. Computational Experiments

The graphs shown below that are produced after the simulations completed.

As it can be seen, they are very important as regards the analysis of the results, as they allow immediately to understand how the system behaved.

The model developed was validated by means of the technique named as face validity that involves domain experts. Through this method the judgment about the accuracy of the model is subjective. In fact, based on experience, it can be said if the model behaves reasonably. In fact, a way for expressing a correct judgment consists to follow the agents properties viewing on a display the behaviour of the model over time (animation method). In all simulations, the number of iterations are equal to 500-1000-1500 (i.e., more or less 18 hours of business activity for each three range). The time range necessary to prepare each document, defined in Table 12. In each simulation the time necessary to prepare the documents is chosen randomly between the values in the range.

Symbol	Description	Range	Unit
ENS	Entry Summary Declaration	[0,1]	-
MMA	Manifesto Merci in Arrivo	[2,40]	M
DO	Delivery Order	[15,240]	M
CDtime	Custom Declaration Time	[1,40]	M
OTtime	Ordine di Transporto (Trasnport Order)	[5,60]	M
GATEINtime	Time needed for contianer to enter to Terminal	[120,480]	M
NOtime	Time needed to produce Nulla Osta	[5,240]	M
PICKUPtime	Time needed to authorize withdrawal	[5,60]	M

Table 12 Time range of variables used in the model

Two different situations are considered:

- the containers imported are immediately processed (case(i)).
- the containers imported have some delay in the import process (case (ii)).

Figures 41,Figure 42, Figure 43, Figure 44, Figure 45 and Figure 46 show the arrival and exit times of containers in the Deliverer agent. These show that some containers arrived earlier than others and they leave later than the next container arrived. The main cause its' stochastic nature of the model, which faithfully simulated situations in the process same as block process, such as excessive delays because of missing or latecomer documents or delayed trucks.

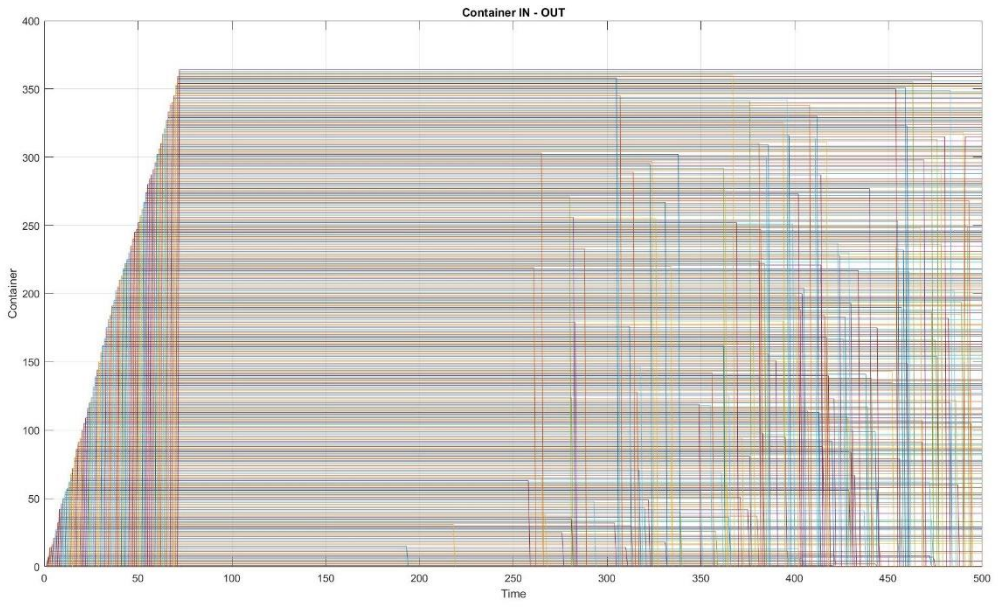


Figure 41 Arrival and exit time of the container in Deliverer agent with 500 its, 10 ens_max and 10 work max

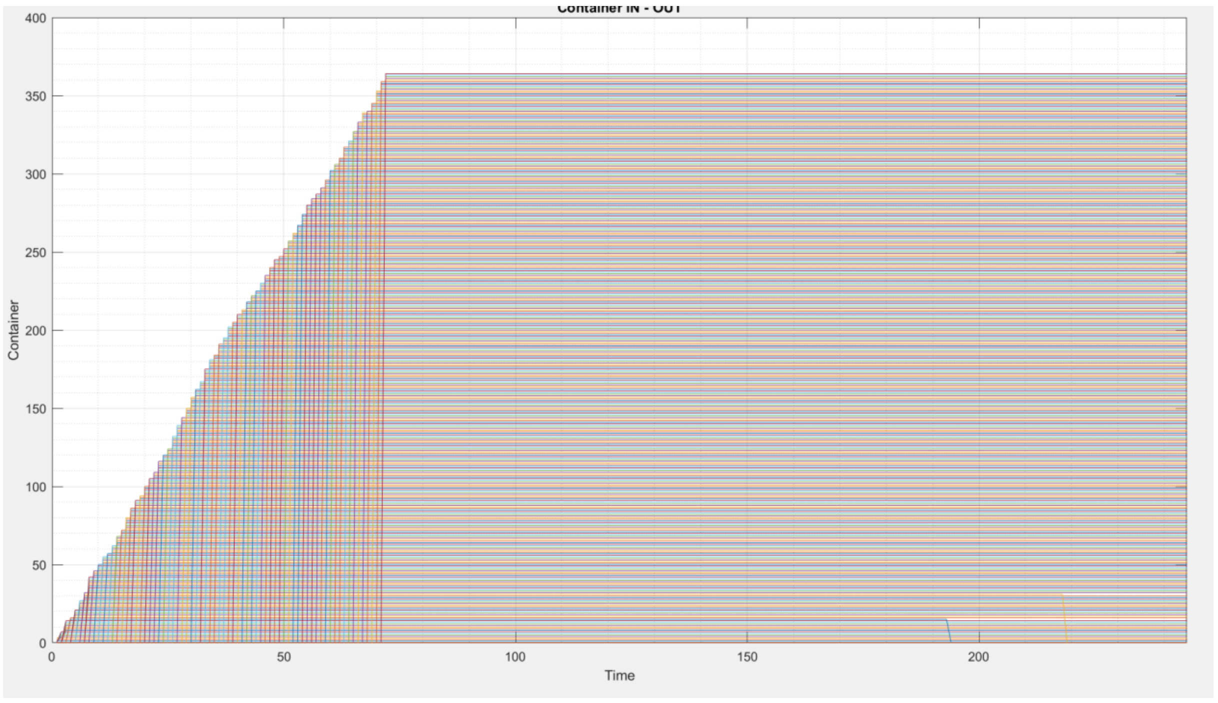


Figure 42 Arrival and exit time of the container in Deliverer agent with 500 its, 10 ens_max and 10 work max Detailed

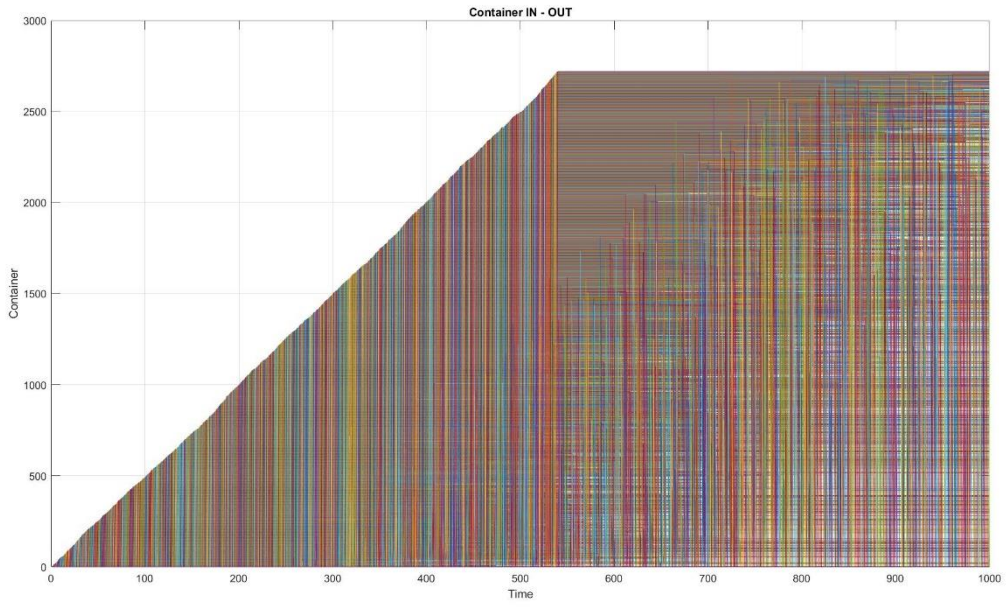


Figure 43 Arrival and exit time of the container in Deliverer agent with 1000 its, 10 ens_max and 10 work max

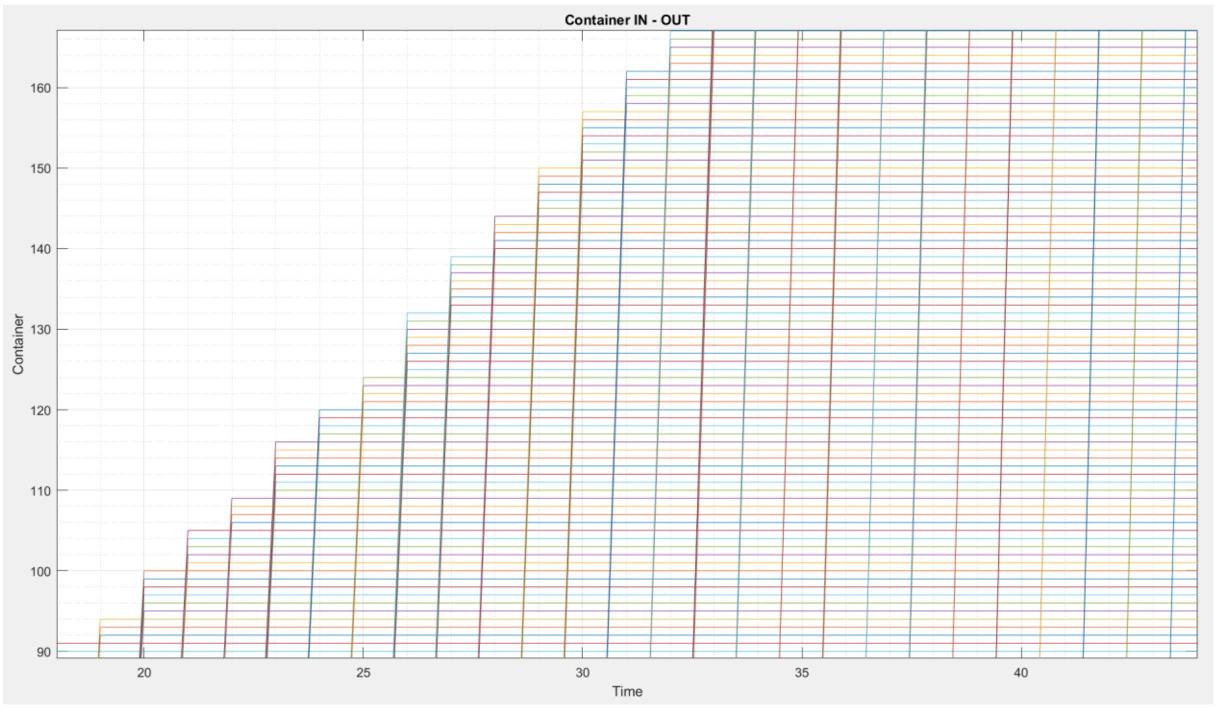


Figure 44 Arrival and exit time of the container in Deliverer agent with 1000 its, 10 ens_max and 10 work max Detailed

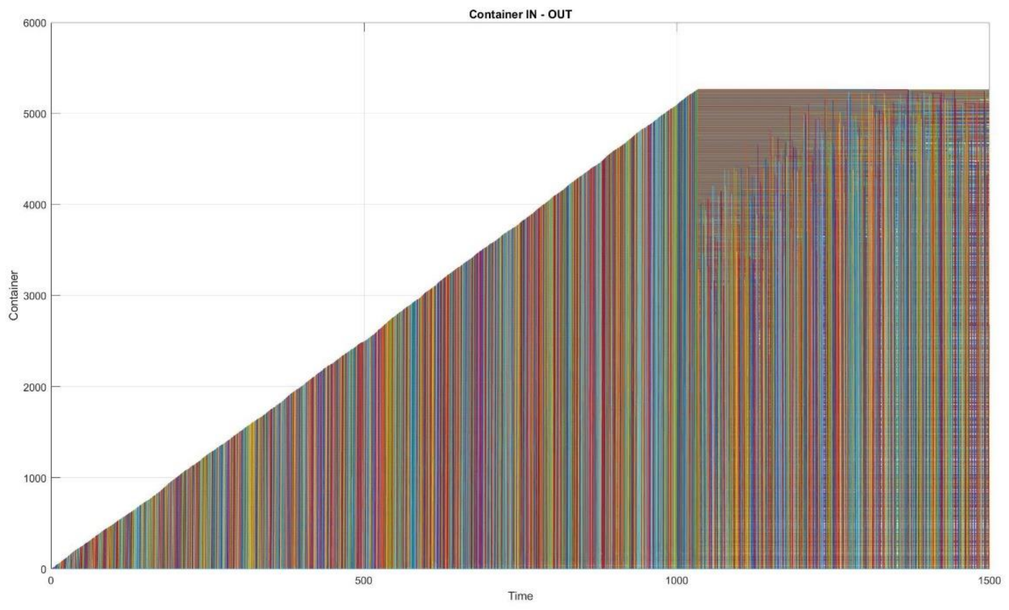


Figure 45 Arrival and exit time of the container in Deliverer agent with 1000 its, 10 ens_max and 10 work max

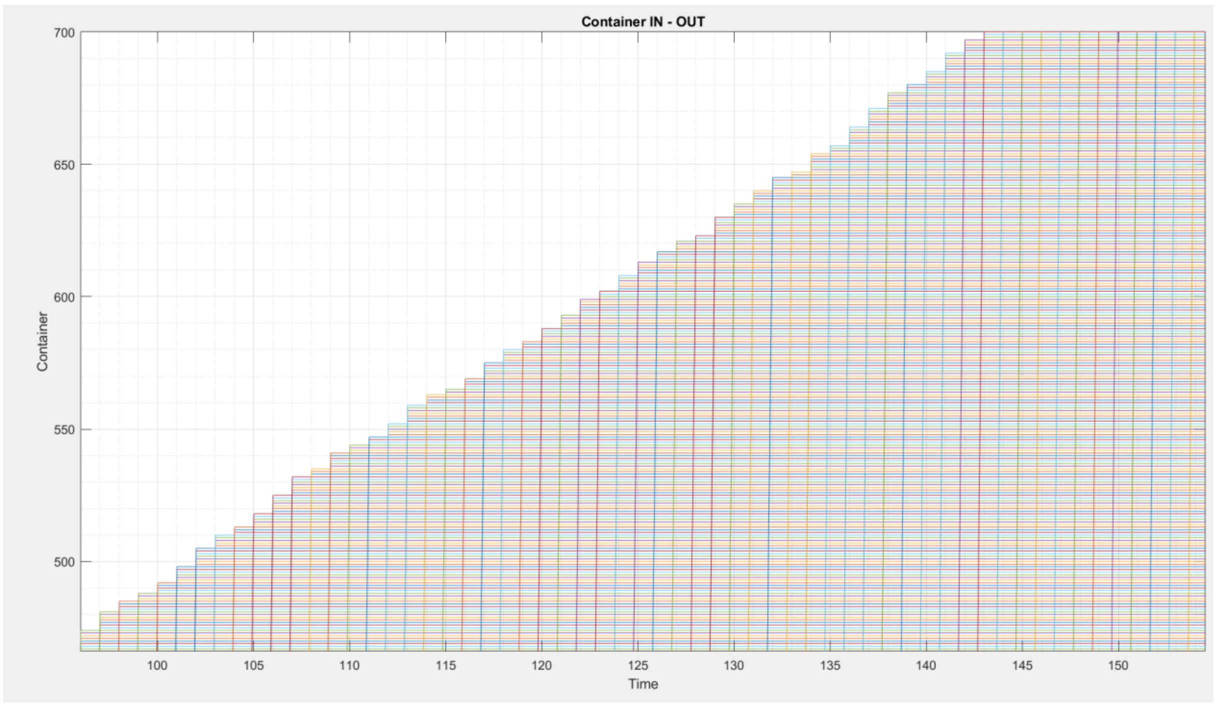


Figure 46 Arrival and exit time of the container in Deliverer agent with 1000 its, 10 ens_max and 10 work max Detailed

Graphs, are shown in Figures 41,Figure 42 and Figure 43, represent the arrival and exit time of the containers in the Deliverer agent (the rising edge indicates the arrival time, the falling edge indicates the exit time) in the case of 500-1000-1500 iterations, ens_max = 10 and work_max = 10: on the abscissas it is present time, on the ordinates the number of the container.

The other graphs of ranges, 500-1000-1500 iterations, ens_max = 10 and work_max = [0,1,2,4,6,8,10] can be seen on List of Graphs section.

In case (i) at each iteration step the maximum number of arriving container is ten, i.e. at each iteration step the number of containers arriving is in the range of [0,1,2,4,6,8,10], whereas in scenario (ii) the maximum number of arriving container is ten, i.e. at each iteration step the number of container arriving is more than the range[0,1,2,4,6,8,10]. Figures 41-42-43 illustrate the containers' arrival and exit time from the port system in scenario (i). It is worth noting that the time for processing a containers from its arrival in the port to its leaving changes showing that if some documents are not ready or are delivered later, the process is blocked. Also the exit time can be seen.

Figure 47 shows the time spent at the port by the containers plotted, showing that the values confirm the right implementation of the model, according to the time ranges provided in Table 12.

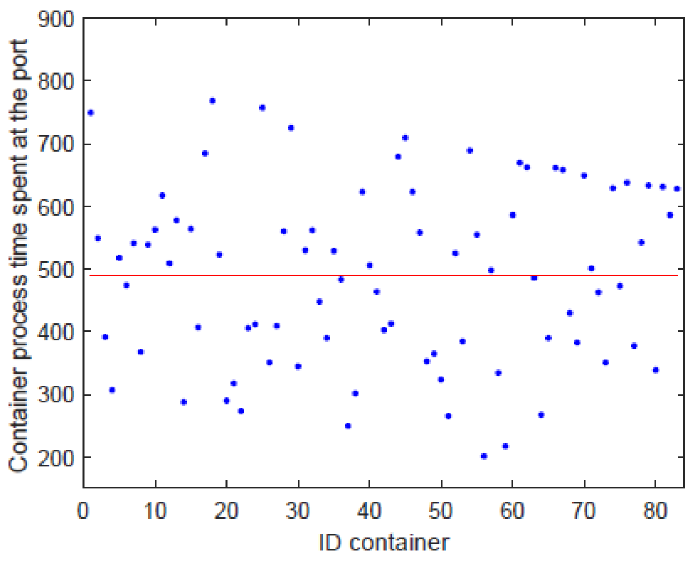


Figure 47 Average of the time and time spent at the port

The blue dots represent the time spent at the port by the first 84 containers in case (i). The red line represents the average time where the iterations are equalized to 800.

Figures 48-49-50 show the number of container contemporaneously presented in the Terminal agent in case (i) and the range of [1,2,4,6,8,10]. In both cases, after a transient situation, where the number of container increases more or less linearly, the regime is received. In the regime the number of containers is varying. It is worth noting that the time employed to process a container differs for each container bringing the oscillatory behavior in the regime part. Comparing the transient part in case(i) and case(ii), it is worth noting that in case (ii) the ramp is steeper because of more container arrive in the same time and the regime is reached faster.

The Figures 48-49-50 show us the number of container contemporaneously presented in the Terminal agent in the case of 500-1000-1500 iterations and ens_max is euqal to 10 and work_max is in the range of [1,2,4,6,8,10].

Other graphs for other 8 agents can be found in the List of Figures section.

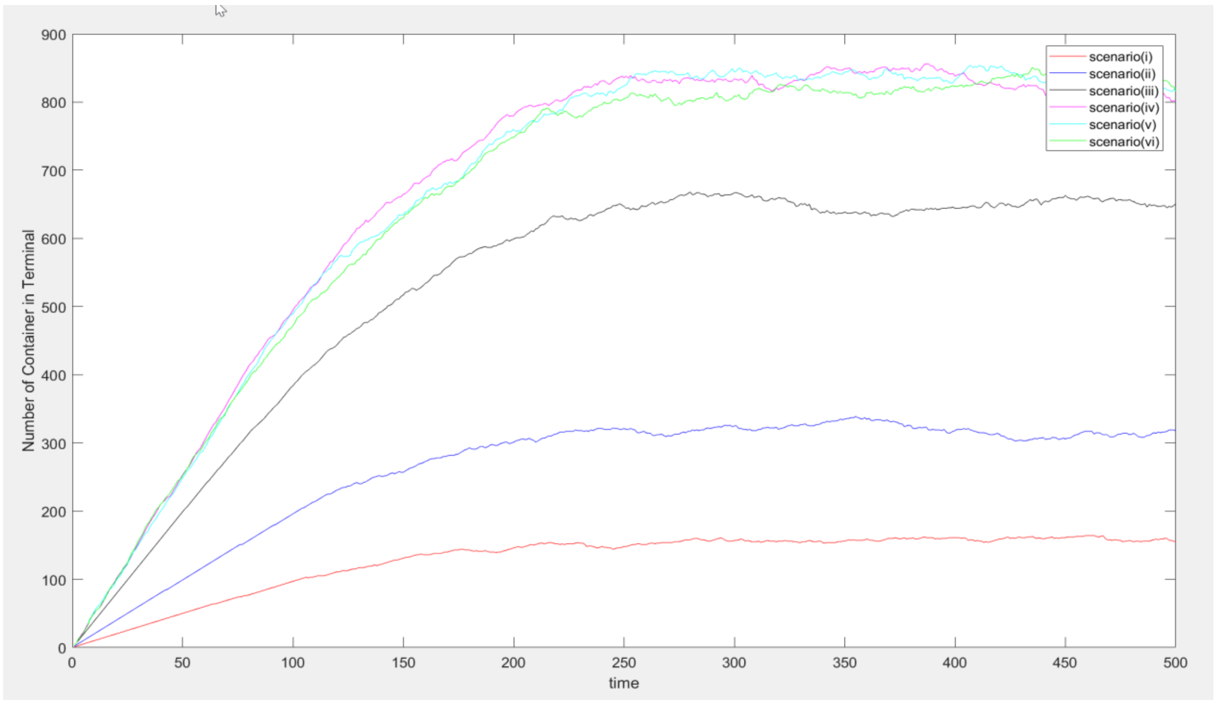


Figure 48 The number of containers in Terminal-500 Iterations

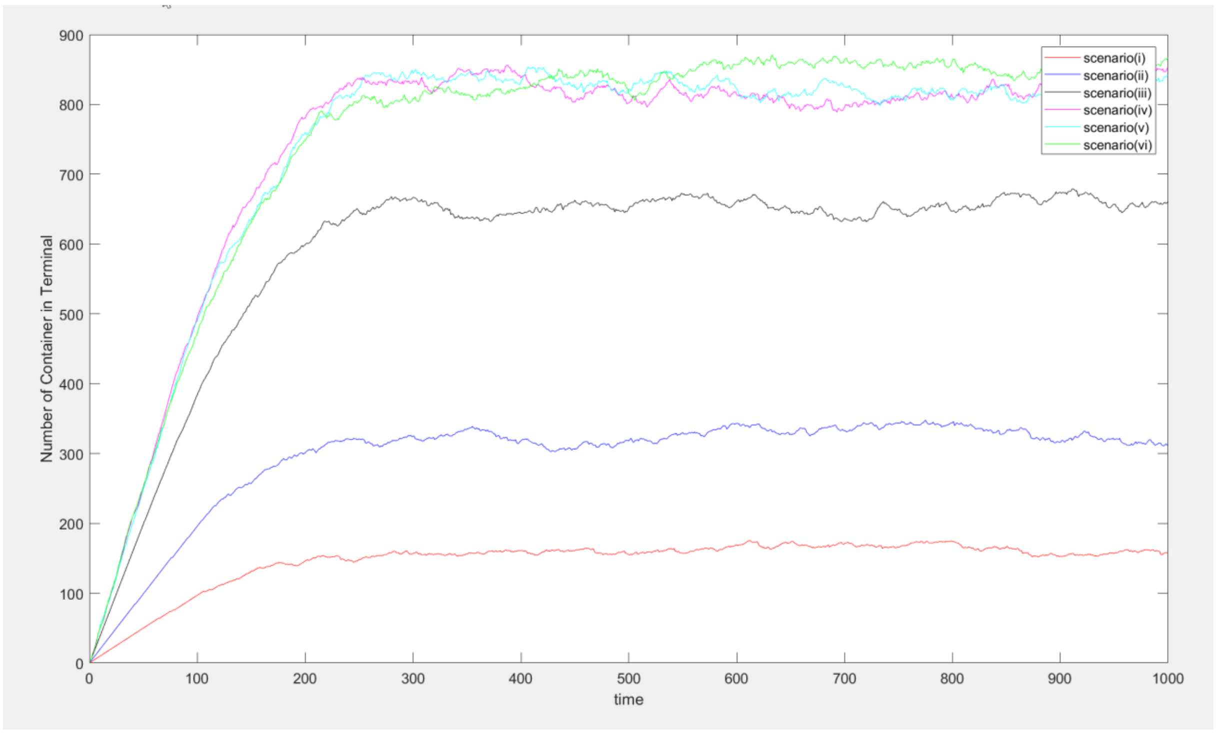


Figure 49 The number of containers in Terminal-1000 Iterations

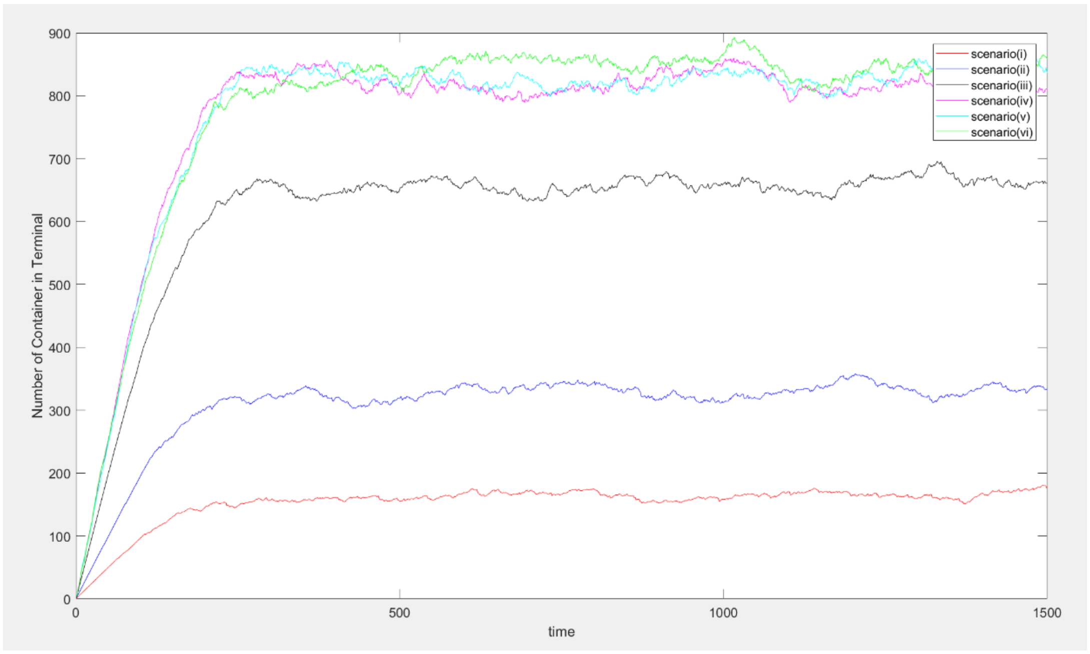


Figure 50 The number of containers in Terminal-1500 Iterations

The figure 51 shows the number of container in queue in the Customs agent in scenario (i) and in scenario (ii), respectively. In the case of scenario (i), red cross line, the number is always equal to zero, as when a container arrives the Customs is ready to process it, whereas in scenario (ii), blue dotted line, if two containers arrive at the same time only one container can be processes and the other is added in the queue. In the simulation performed, the queue is managed according to the First In First Out (FIFO) logic.

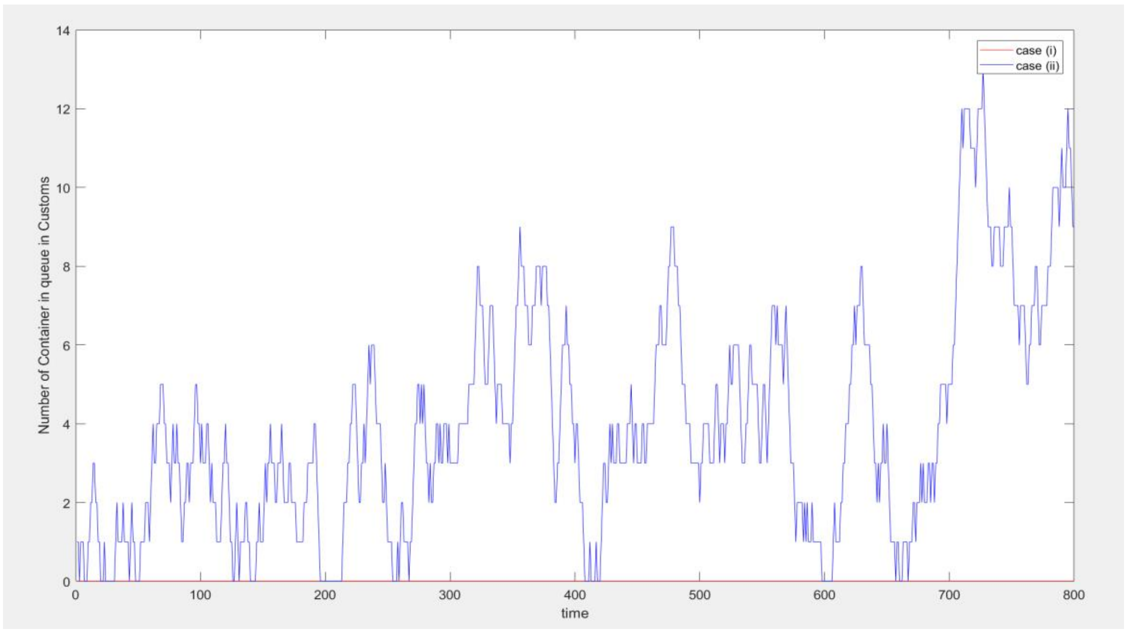


Figure 51 Number of container in queue in the Customs agent-800 its sample

Figures that are showed in starting from figure 51 allow us to understand where bottlenecks are, i.e. the accumulation of containers, so as to identify them easily and then analyze the data which is important.

Graphs 48-49-50-51 show the number of containers in the queue in Customs Agent and it is produced with the values of ens_max and customs_work_max, if they are different, it requires that Customs have to put in queue of containers because it can not process them. It allows to understand what are the points of greatest influx of containers?

The figures 51 show the number of container in queue in the Customs agent in case (i) and in case (ii), respectively. In the case of case (i), red cross line, the number is always equal to zero, as when a container arrives the Customs is ready to process it, whereas in scenario (ii), blue dotted line, if two containers arrive at the same time only one container can be processes and the other is added in the queue. In the simulation performed, the queue is managed according to the First In First Out (FIFO) logic. These are the same for Figure 52-53-54, the difference is iteration numbers.

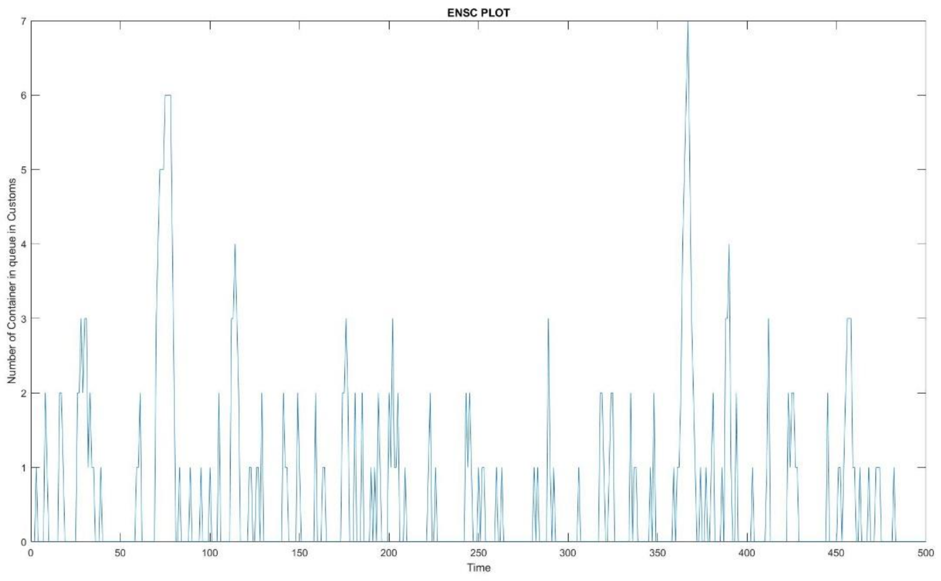


Figure 52 Number of container in queue in Customs-500 iterations

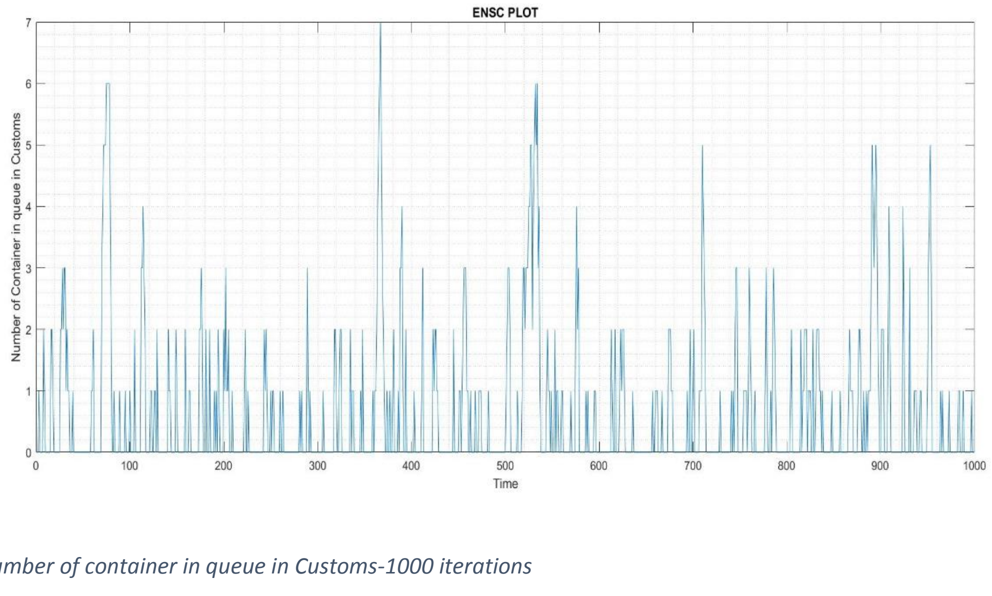


Figure 53 Number of container in queue in Customs-1000 iterations

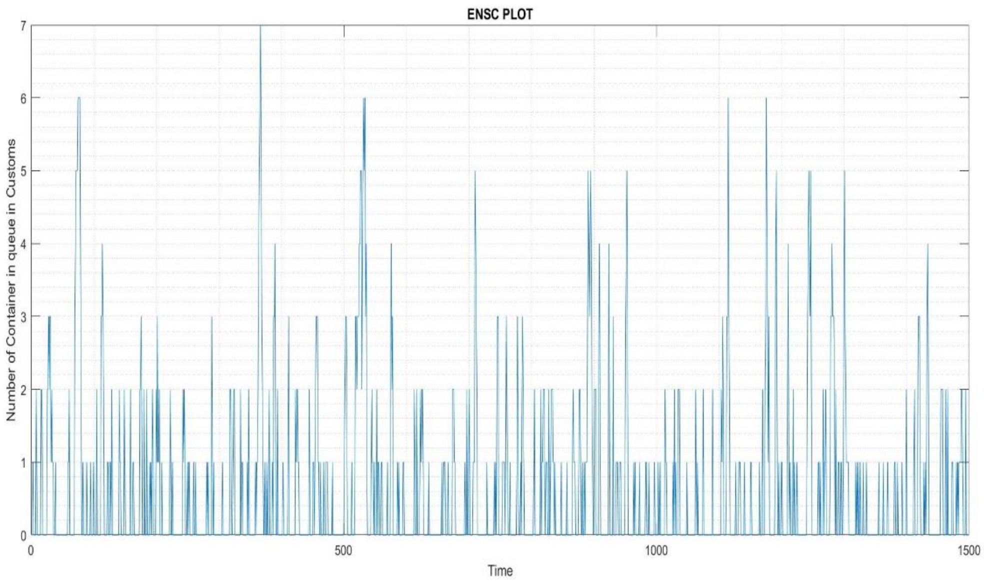


Figure 54 Number of container in queue in Customs-1500 iterations

On the other hand, in the scenario for 500-1000-1500 iterations, the graph will change. As it can be seen in the Figures 55-56-57, scenario VI are similar with 800 iterations version of the graph. But the other ranges are various than the earlier one. These graphs are combined and shown with the all ranges of work_max [1,2,4,6,8,10] in Figures 55-56-57.

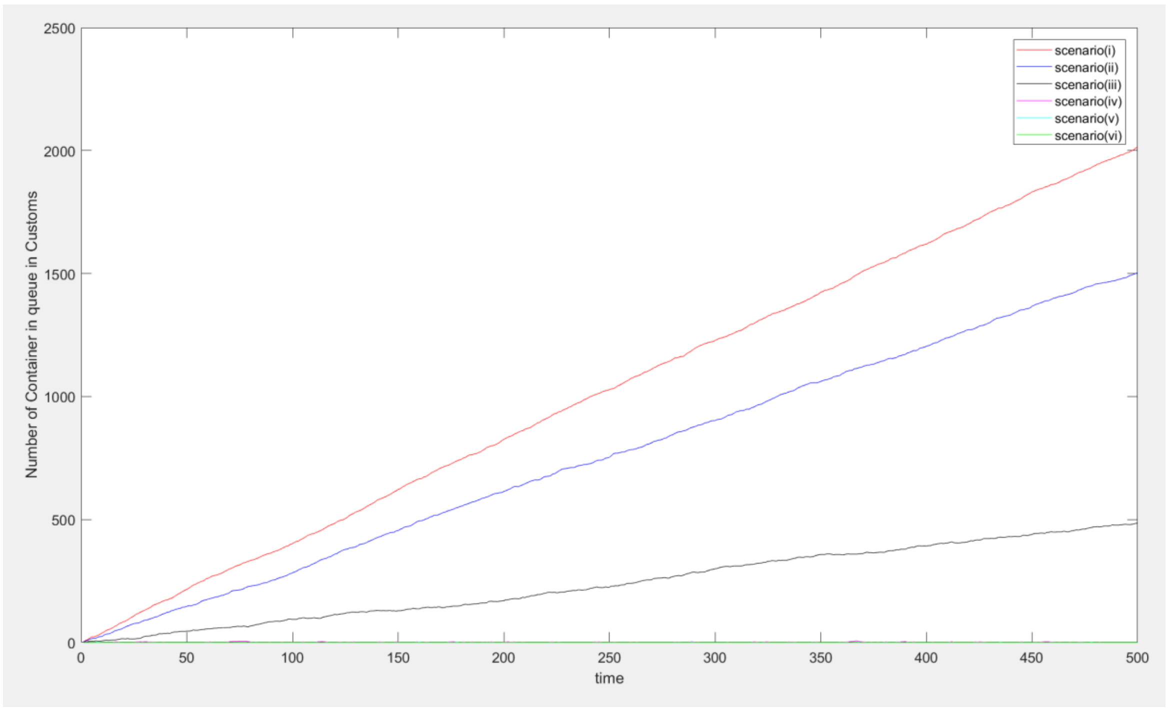


Figure 55 Number of container in queue in the Customs agent-500 ens_max 10 and work_max [1,2,4,6,8,10]

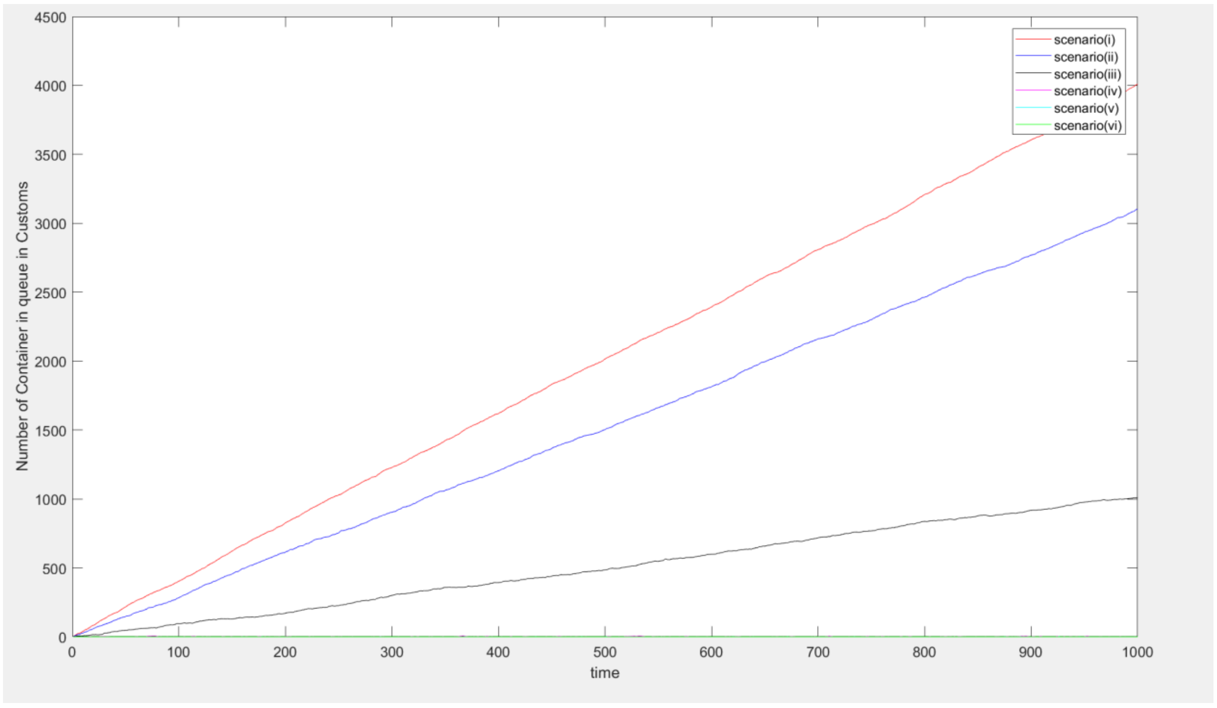


Figure 56 Number of container in queue in the Customs agent-1000 ens_max 10 and work_max [1,2,4,6,8,10]

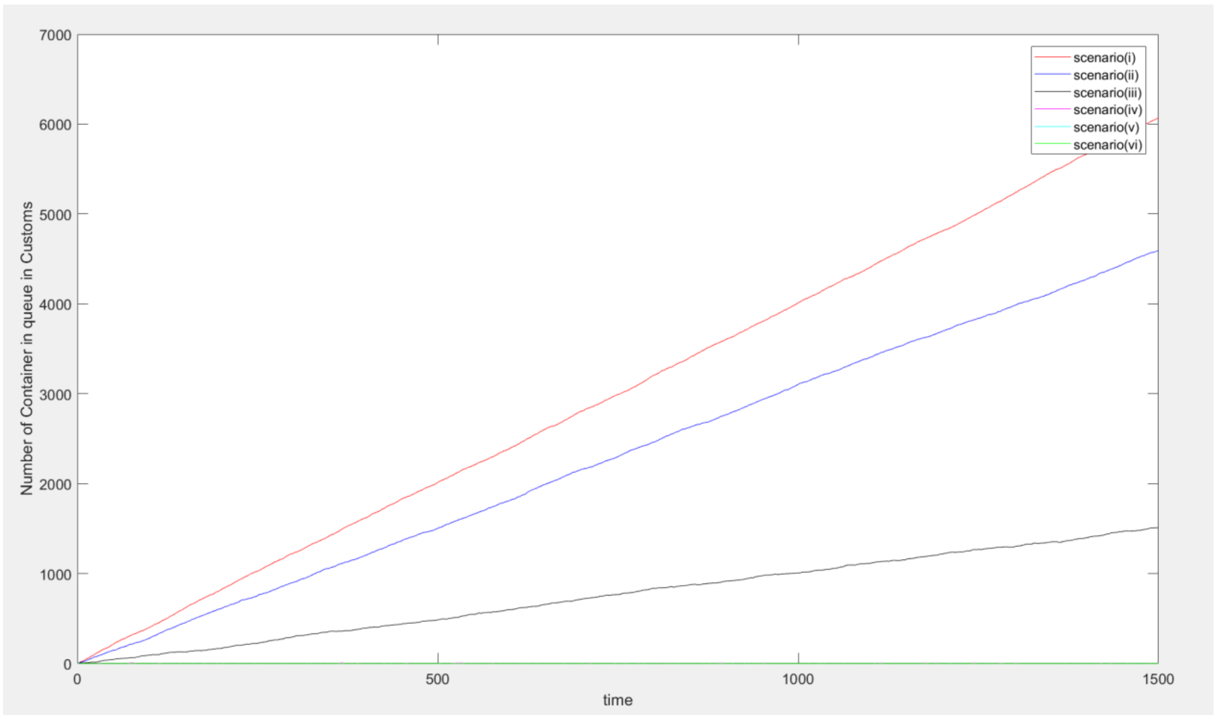


Figure 57 Number of container in queue in the Customs agent-1500 ens_max 10 and work_max [1,2,4,6,8,10]

In particular in the types of Graphs 58-59-60 a transition is observed, its' length depends on the agent and the parameters that are set in the simulation, which identifies a zone, where the number of containers inside, it grows almost linearly and a second zone that can be defined "regime", in where the number stabilizes and fluctuates. This behavior similar to second-order system is due to the fact that used time, the number of arrival and exit containers. In addition, it is assumed that the system in the start state is "empty", or without the presence of previously arrived containers. In addition, sizeend is equal to containers that arrive to the agent.

Therefore, if it is desired to see the situation of the system at full capacity, it will be necessary to execute longer simulations (for example from at least 1800 iterations), initial part is excluded.

Graphs represent the variable sizeend of the Terminal agent, i.e. the number of containers within the aforementioned agent, in the case of 500-1000-1500 iterations, ens_max = 10 and work_max = 10. The other graphs for 500-1000-1500 iterations, ens_max = 10 and work_max = 10

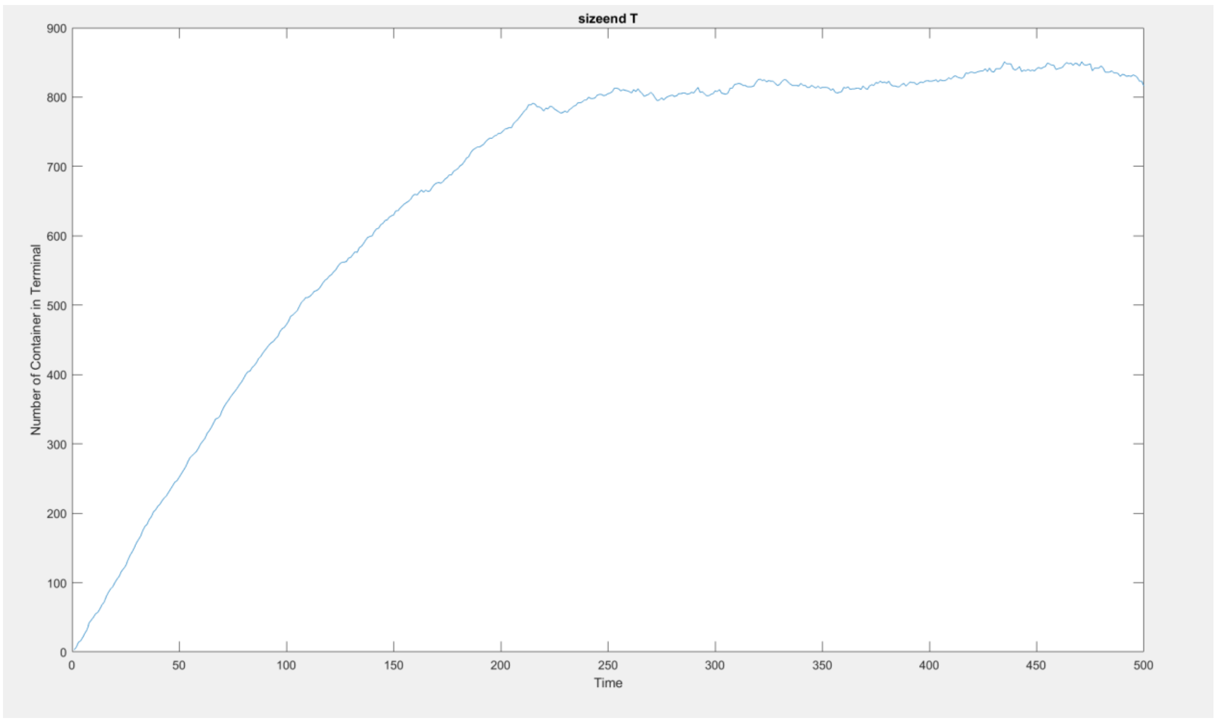


Figure 58 Sizeend of the Terminal Agent- 500 its ens_max 10 and work max 10

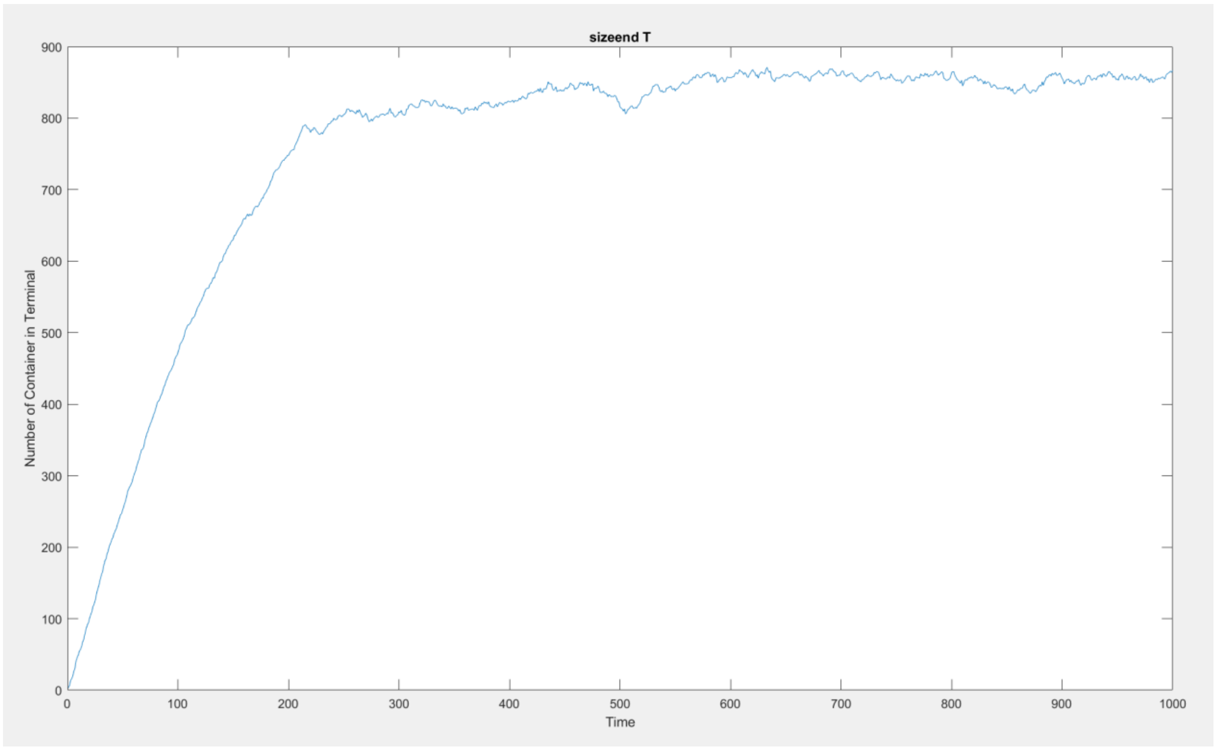


Figure 59 Sizeend of the Terminal Agent- 1000 its ens_max 10 and work max 10

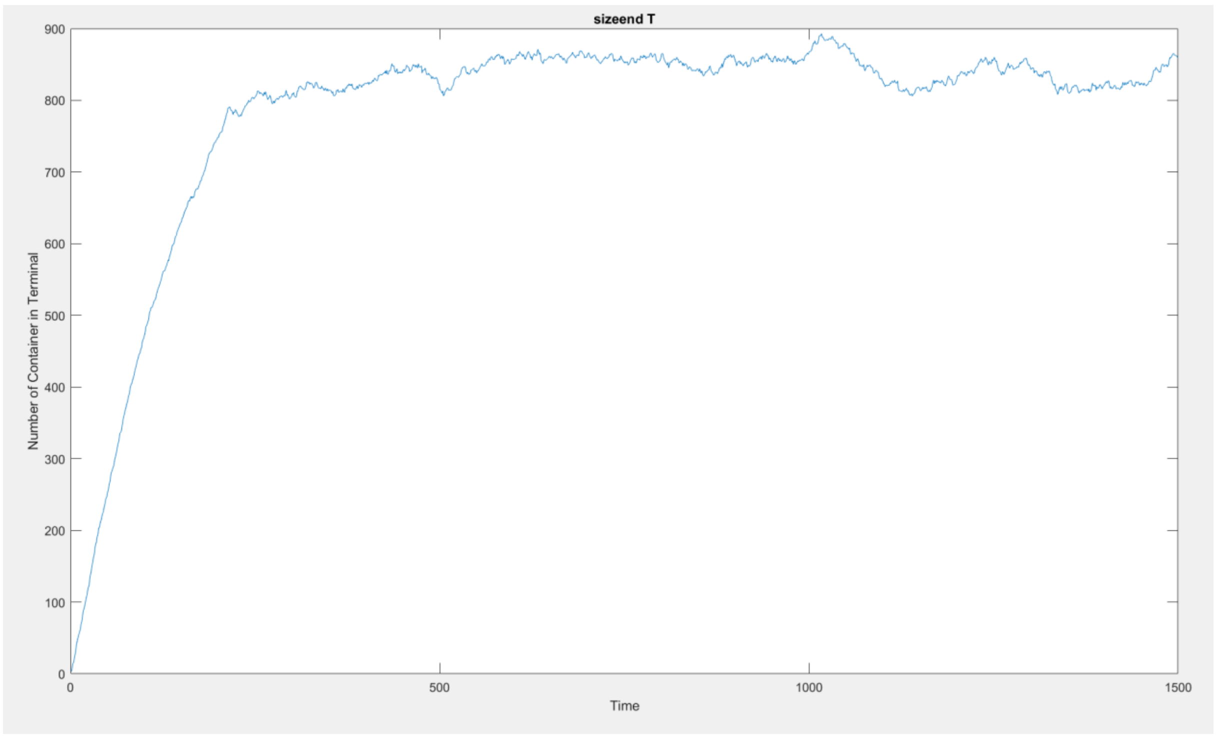


Figure 60 Sizeend of the Terminal Agent- 1500 its ens_max 10 and work max 10

The Graph 61 shows containers that are in the queue. FIFO policy is used for the disposal of containers, the behavior is straight because the number of the container in the queue will be that arrived later. However, it is possible to modify this policy, for example, a priority can be given to some containers, it may happen that some containers are made to wait longer than others.

It is worth remembering that in scenario (i) there are no container in the queue, whereas in scenario (ii), as the logic used is FIFO the curve is an ascending straight line, as shown in Figure 48. Thus, the import process of goods implemented works and let to investigate the main bottlenecks of the system.

In fact, the two situations tested helps us to consider different what-if scenarios and to understand the level of criticality of the network and the nodes and where the mechanism does not work, generating bottlenecks which slow down the process flux.

In the Graph 61, Graph 62, it can be seen if the work_max increase, containers in the queue will decrease.

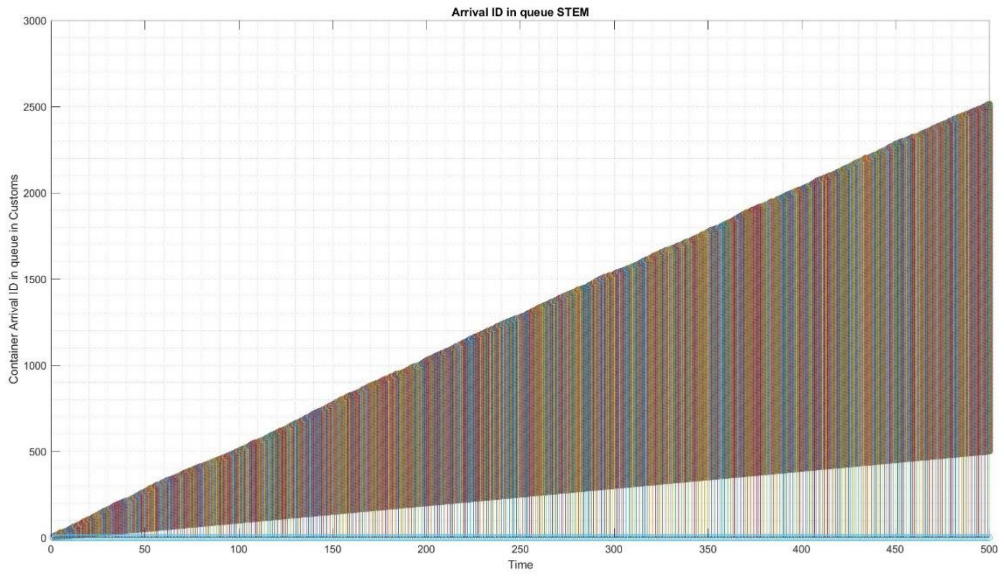


Figure 61 Sequence of container in the Customs queue – 500 iterations ens_max 10 and work max 1

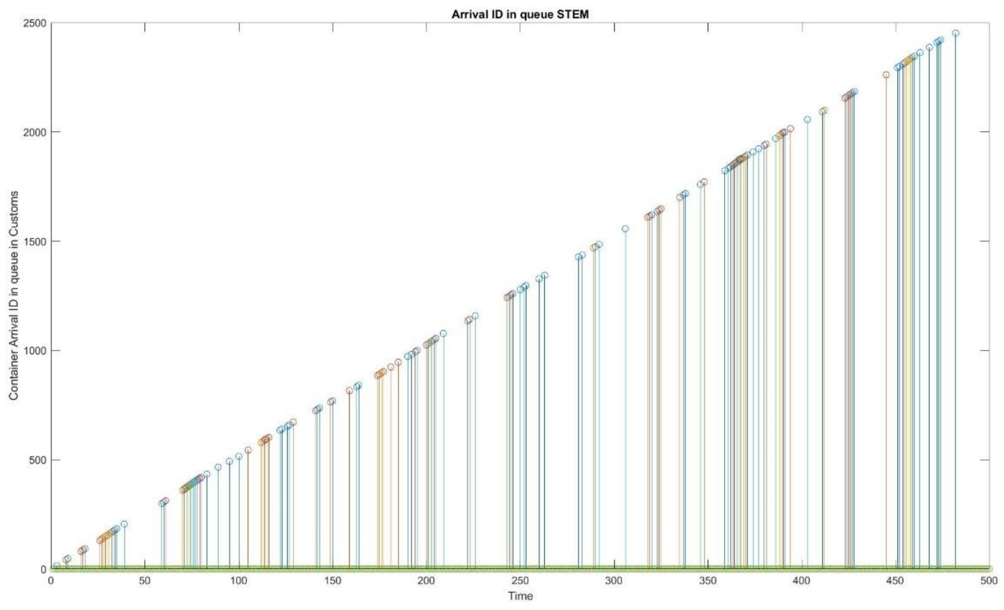


Figure 62 Sequence of container in the Customs queue – 500 iterations ens_max 10 and work max 6

It can be seen the difference between work_max = 1, work_max = 6 and work_max = 10 where the ens_max is always equal to 10. In the case of work_max = 10, there is no any container in the queue as it is expected.

Moreover, the graphs for 1000 iterations and 1500 iterations can be seen in the List of Figures part.

It is worth saying that 272 graphs are produced. As it can be imagined that it is not possible to show all of them. But they can be shown in case of request.

Concluding remarks

This thesis focuses on the development process of the simulator for the import and clearance processes of containers at the Port of Genoa and it investigates the applicability of ABMSs in supporting operational management of port traffics to improve the performance of the whole port system.

The objectives are; to create a computational environment to make useful experiments to identify the causes of possible bottlenecks within the process, to provide a simulation that can identify the paths of individual containers, and to observe the time and behavior with the authorization for each one.

The import process of the goods has been modelled and implemented for simulating the management processes in order to identify local practices made by the different actors involved in the activities. The computational environment developed has been tested with the real case of the port of Genoa. The model chosen for the study falls into the category of agent-based models which are very suitable for simulating complex systems where a large number of heterogeneous agents or entities with specific features interact among each other in order to establish relationships. Agents based modeling has enabled us to investigate the dynamics of the Port of Genoa and its macroscopic behaviour starting from the micro-level properties, constraints and rules assigned to selected agents. Furthermore, agent-based modelling has been used to study pragmatic solutions to problems related to business processes. The software chosen for the implementation is FLAME.

We have set the model according to the structure of the agent based model because it is the one that most closely matches the port environment, as there are many different actors acting individually and collectively, because they communicate with each other easily.

The model was produced using the FLAME framework and specifications provided by the experts at the port and its environment (workers and researchers).

Many variables have been defined various variables in stochastical approach to maintain the unpredictability behavior of the port, such as the variability of loading times and production of documents or the arrival or picking up of containers.

Precisely for this reason, it was thought that there could be problems such as the most important reasons are the hitches called "bottlenecks", which are slowdowns and stack of processes because of the stack of delays in the processes. Others were the lack of determinated authorizations at determinated times, which also cause delays.

The results are satisfactory because through the elaborated products that are the databases and the graphs, it can be clearly understood the functions of the processes.

In particular, the graphs of the number of containers inside the agents help to identify the location of the slowdowns of the process, and through the databases it is easy to control each individual data in order to the path of the all containers minutely.

This application is included to the GESTEC project and the developed software framework guides to manage complex organizations with particular connection to the port area.

Precisely for this reason the application is developed so that it is able to handle other cases, such as any dangerous goods (illegal or dangerous containers), which must be processed separately to avoid unpleasant issues, or priority of containers, which require faster customs clearance (contains perishable goods), or the integration of determinated agents are able to intervene in the event of influx.

The business case of the port of Genoa has been tested modeling the time documentation according to the specification of the Genoa case.

Results have shown that the mechanism implemented simulates the actual process. Moreover some bottlenecks have been discovered, such as delays to the handling of the containers and queues formation due to missing documentation or documentation with errors or not ready. Thus, the next step will be to use the implemented simulator to perform computational experiments that help to investigate and find solution for these and other possible bottlenecks in the import process, following the behaviour of the single container. Furthermore, containers with different priority will be implemented, considering the case of dangerous goods or perishable goods. Finally, also different techniques for the model’s validation will be considered. If the empirical data, i.e. data about the flux of containers, will be available, than, for example, the sensitivity analysis will be considered.

The main contribution of the work concerns a simulation tool for evaluating management policies in activities flow of a port. In fact the model of the port community and its assets is populated by many agents (stakeholders) that have individual goals (set of functions that are specified). The trade-offs found at a specified state can be changed through simulation experiments. The set or ranges of parameters can be evaluated while the simulation permits to compare several alternatives, i.e. supporting stakeholder relations management.

We have presented the initial steps in developing an ABMS of the import process of goods in the port of Genoa. The goal is to develop an ABMS that can be used for evaluating policies for port terminal systems from stakeholders views. The concepts underlying the model can also be used to analyze relations of actors involved on a broader scale, i.e. the port hinterland. As such the FLAME provides an effective tool to structure stakeholder relations and as such is helpful when developing a more structured stakeholder relations management. The ABM developed would be the basis for a decision-support system.

The results of the simulation are not an optimum policy solution, but provide to the decision makers the capacity to view the structure of a port system and the functions that the actors involved have based on several ”what if” analyses.

Appendix

Attached is the "stategraph_colour.pdf" previously introduced in the Stategraph section and as a whole stategraph can be seen on the another pdf document that will be given with the thesis.

As an example, the code for the "AgentMaritime_send_ens" function is shown:

```
int AgentMaritime_send_ens() /* It's an output function. If ENS_MAX is different to 0, send
all the elements in ENS_LIST at Customs with the message agentmaritime_ens_customs.
Then, reset the ENS_LIST array. */
{
  int i;

  if (ENS_MAX != 0)
  {
    for (i = 0; i < ENS_LIST.size; i++)
    {
      ENS1 = ENS_LIST.array[i];
      add_agentmaritime_ens_customs_message(ENS1); // Message sent

/* Here we check the environment constant "debug_mode": if it's 1, print the following
document. */
      if (DEBUG_MODE)
      {
        FILE *file1;
        char *filename;
        filename = malloc(100 * sizeof(char));
        filename[0] = 0;
        strcpy(filename, "its/AgentMaritime_send_ens_1.txt");
        file1 = fopen(filename, "a");
        if (file1 == NULL)
          printf("Agent Maritime send ens %s \n", filename);
        fprintf(file1,"\n %d %d %d %d %d", MIN, ID, ENS, ENS1,
ENS_LIST.size);
        fclose(file1);
        free(filename);
      }
    }

    if (DEBUG_MODE)
    {
      FILE *file1;
      char *filename;
      filename = malloc(100 * sizeof(char));
      filename[0] = 0;
      strcpy(filename, "its/AgentMaritime_send_ens.txt");
      file1 = fopen(filename, "a");
      if (file1 == NULL)
        printf("Agent Maritime send ens %s \n", filename);
      fprintf(file1, "\n %d %d %d %d", MIN, ID, ENS, ENS_LIST.size);
      fclose(file1);
      free(filename);
    }
  }
  reset_int_array(&ENS_LIST);
  return 0;
}
```

List of Graphs

Please consider that, there are 272 graphs in a various ranges of ens_max and work_max with 500-1000-1500 iterations. Only some examples of graphs are shown on the list of Graphs part.

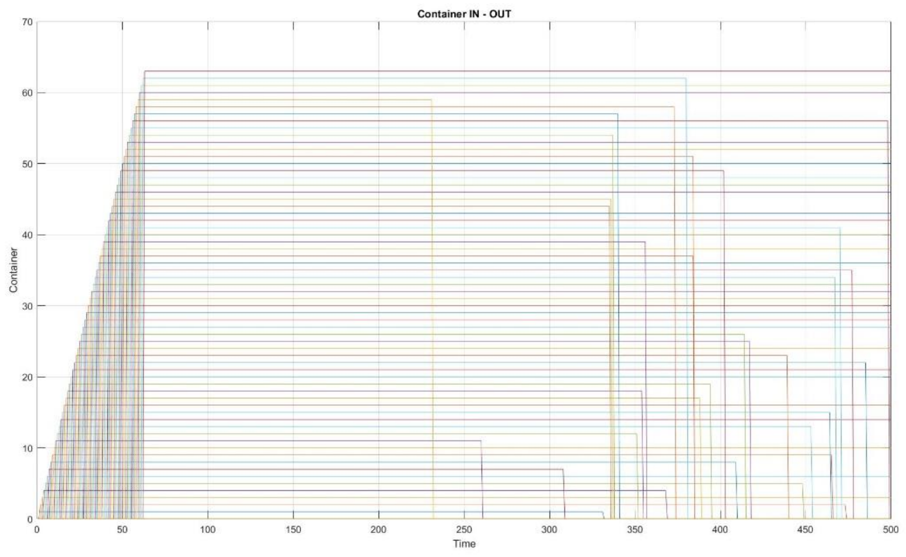


Figure 63 Arrival and exit time of the container in Deliverer agent with 500 its, 10 ens_max and 1 work max

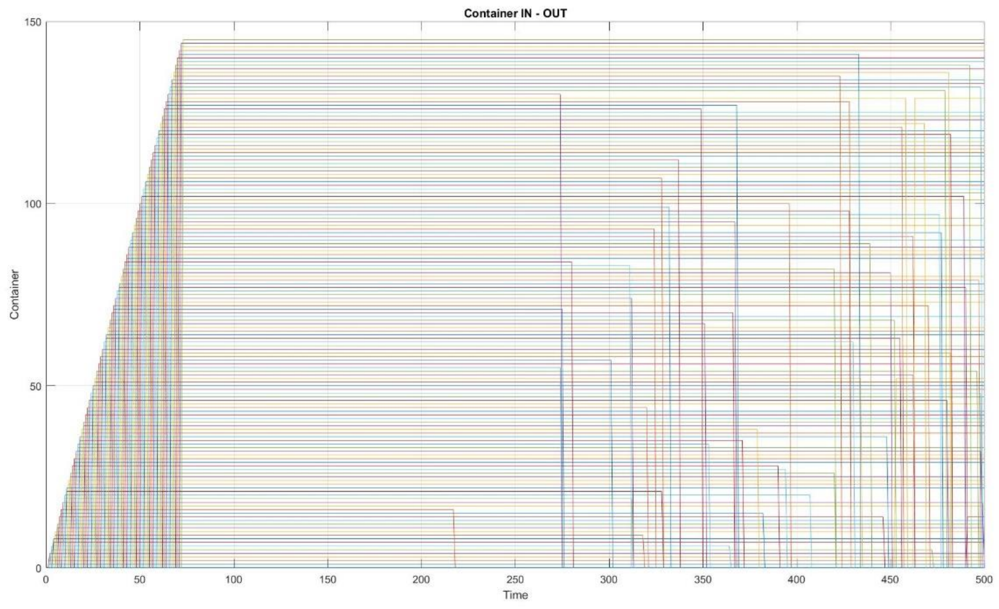


Figure 64 Arrival and exit time of the container in Deliverer agent with 500 its, 10 ens_max and 2 work max

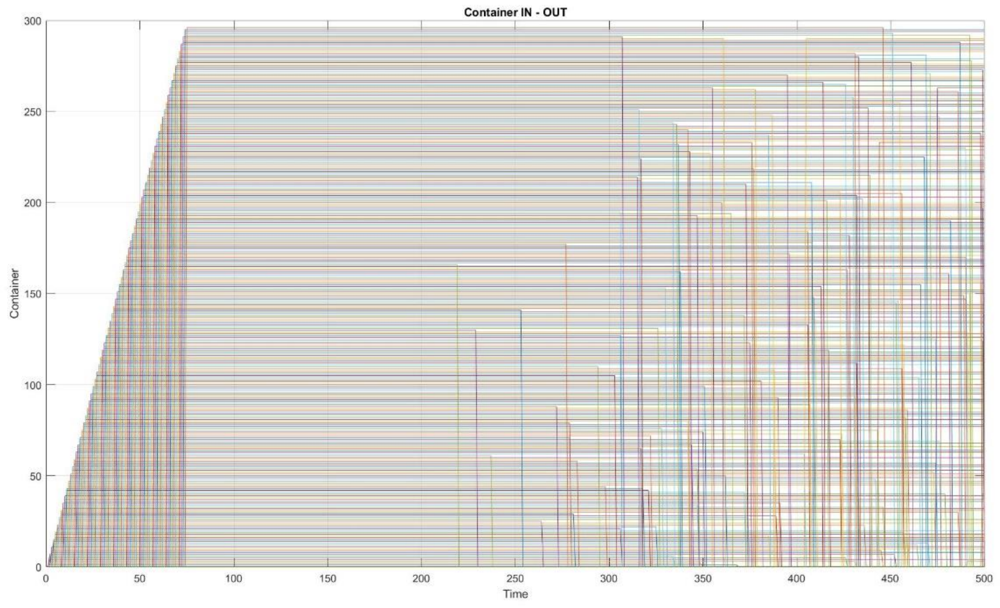


Figure 65 Arrival and exit time of the container in Deliverer agent with 500 its, 10 ens_max and 4 work max

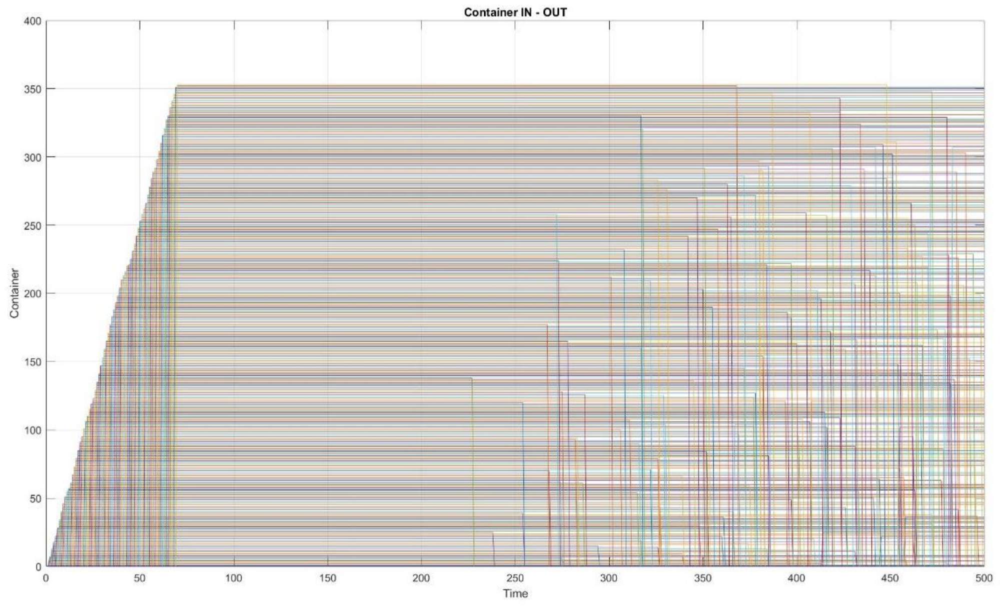


Figure 66 Arrival and exit time of the container in Deliverer agent with 500 its, 10 ens_max and 6 work max

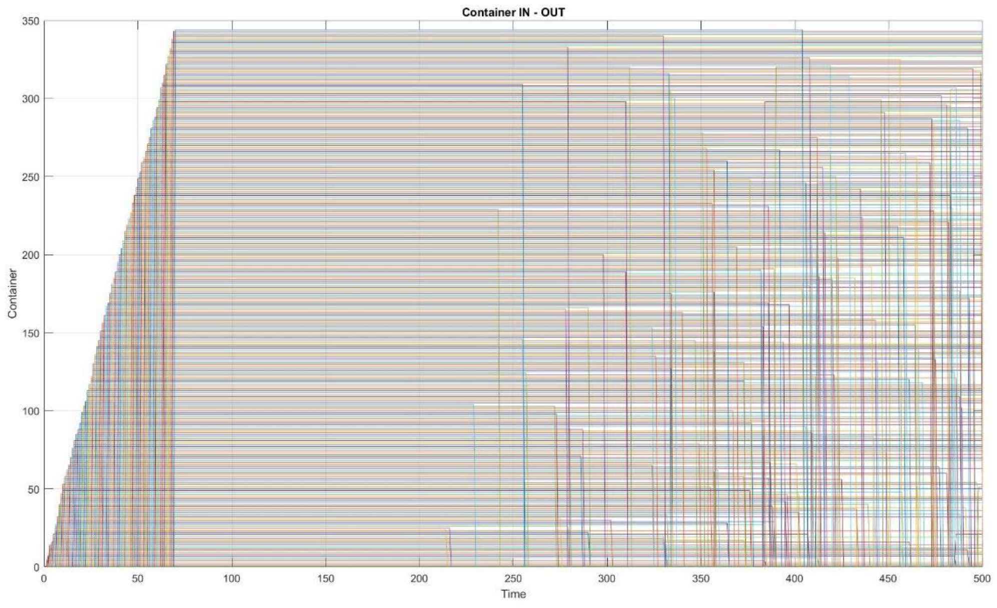


Figure 67 Arrival and exit time of the container in Deliverer agent with 500 its, 10 ens_max and 8 work max

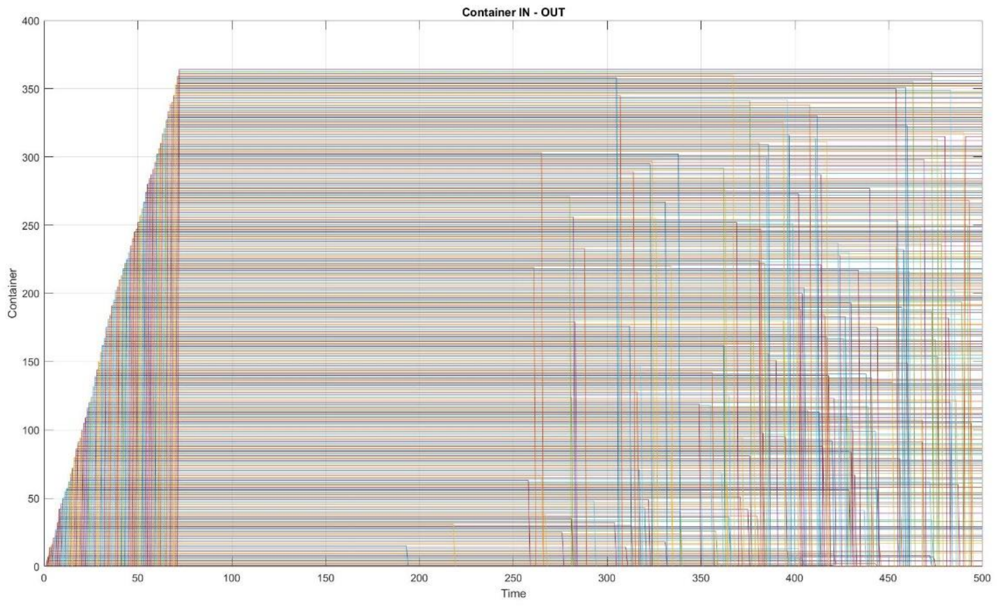


Figure 68 Arrival and exit time of the container in Deliverer agent with 500 its, 10 ens_max and 10 work max

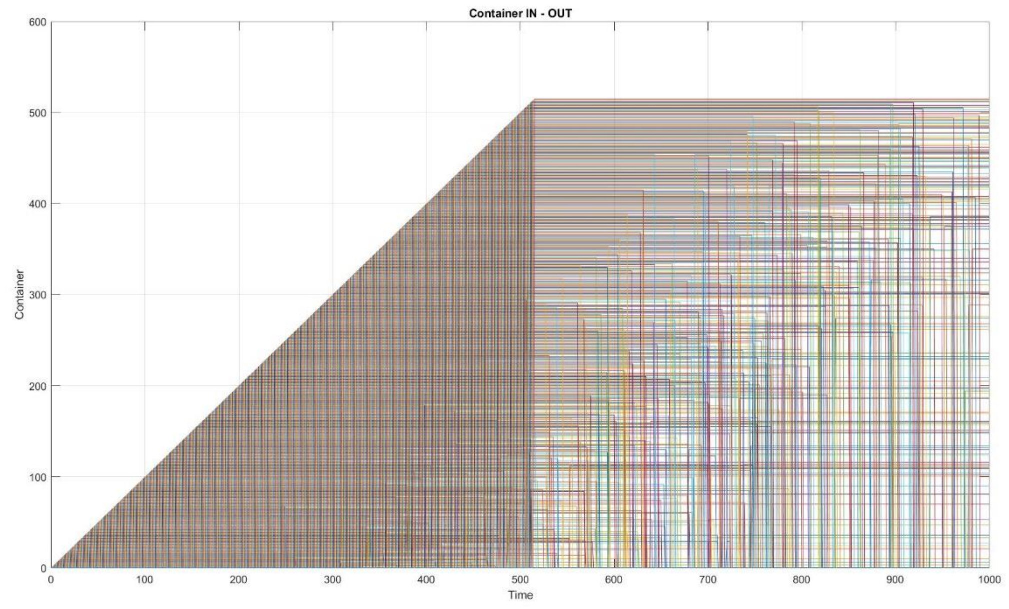


Figure 69 Arrival and exit time of the container in Deliverer agent with 1000 its, 10 ens_max and 1 work max

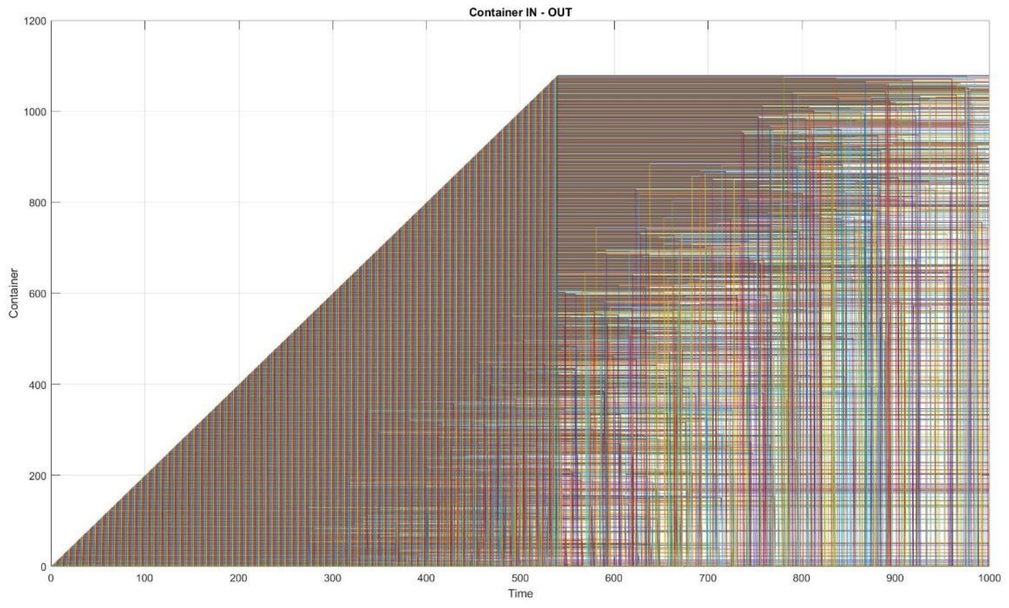


Figure 70 Arrival and exit time of the container in Deliverer agent with 1000 its, 10 ens_max and 2 work max

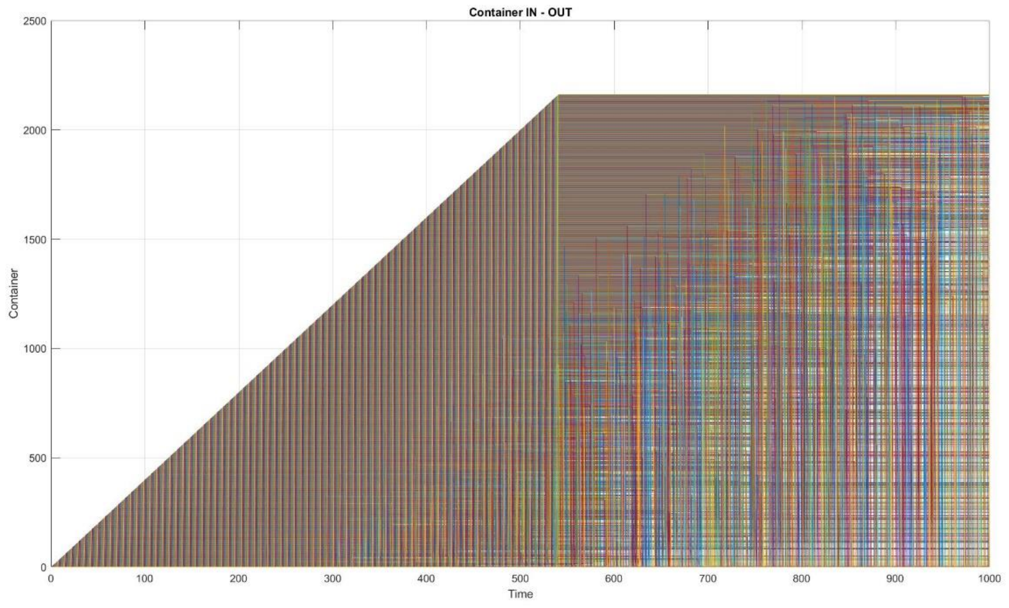


Figure 71 Arrival and exit time of the container in Deliverer agent with 1000 its, 10 ens_max and 4 work max

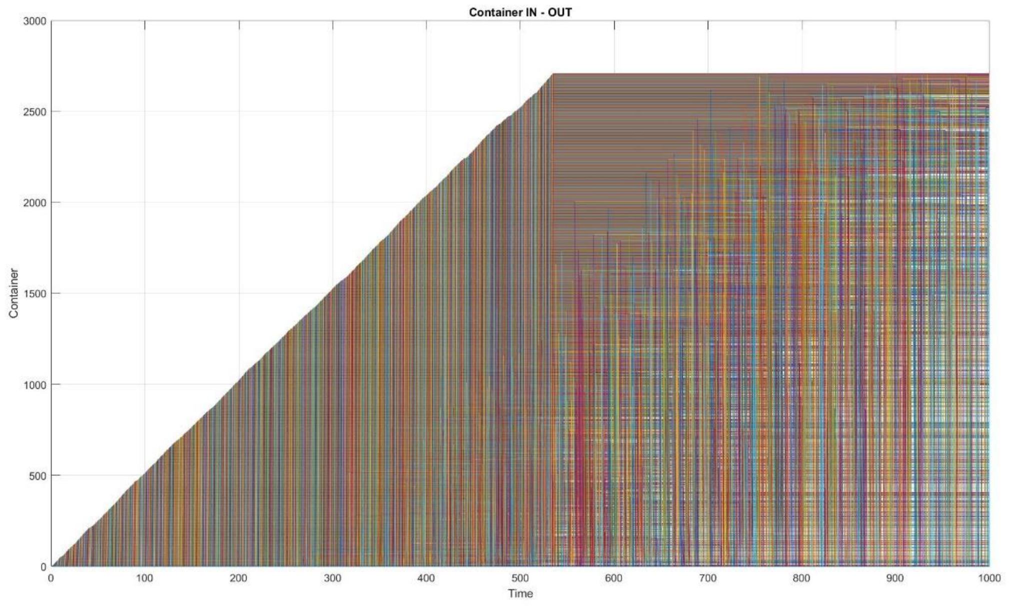


Figure 72 Arrival and exit time of the container in Deliverer agent with 1000 its, 10 ens_max and 6 work max

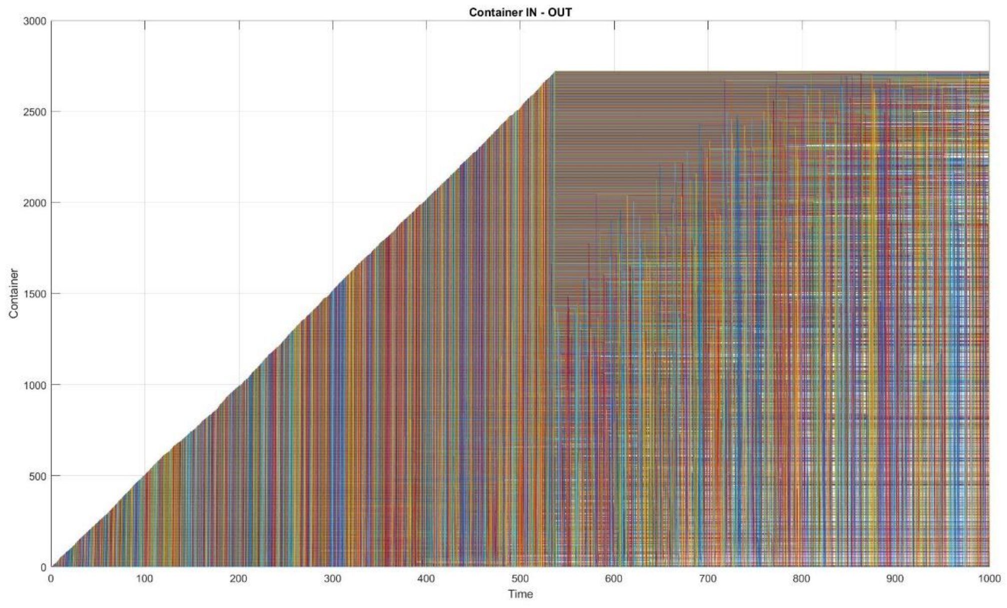


Figure 73 Arrival and exit time of the container in Deliverer agent with 1000 its, 10 ens_max and 8 work max

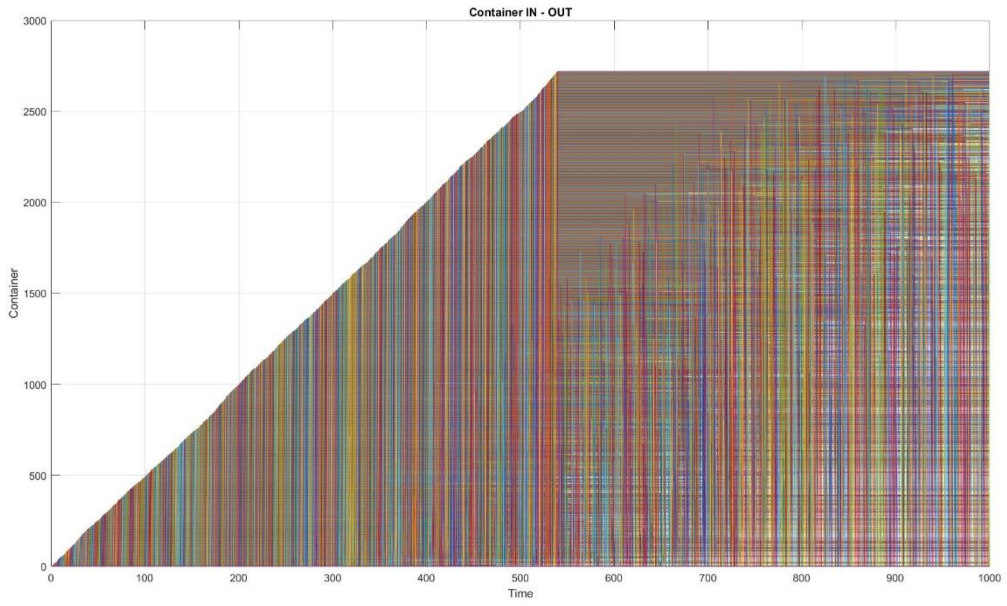


Figure 74 Arrival and exit time of the container in Deliverer agent with 1000 its, 10 ens_max and 10 work max

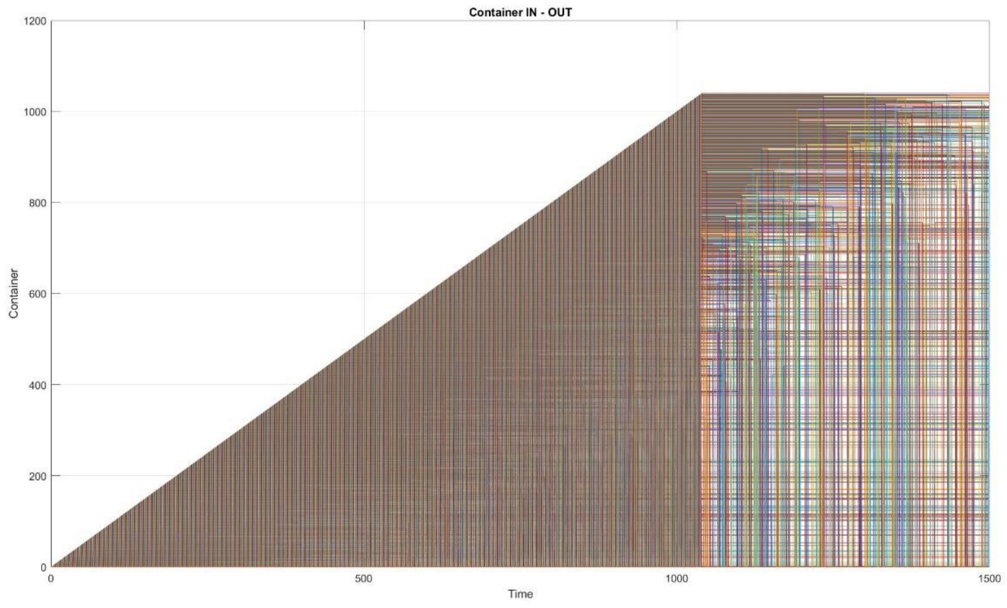


Figure 75 Arrival and exit time of the container in Deliverer agent with 1500 its, 10 ens_max and 1 work max

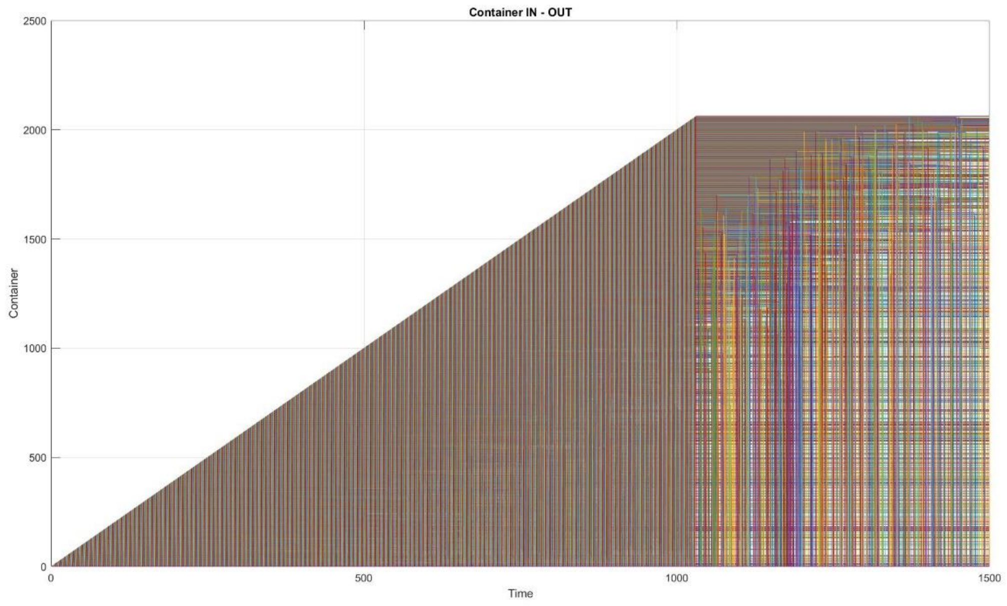


Figure 76 Arrival and exit time of the container in Deliverer agent with 1500 its, 10 ens_max and 2 work max

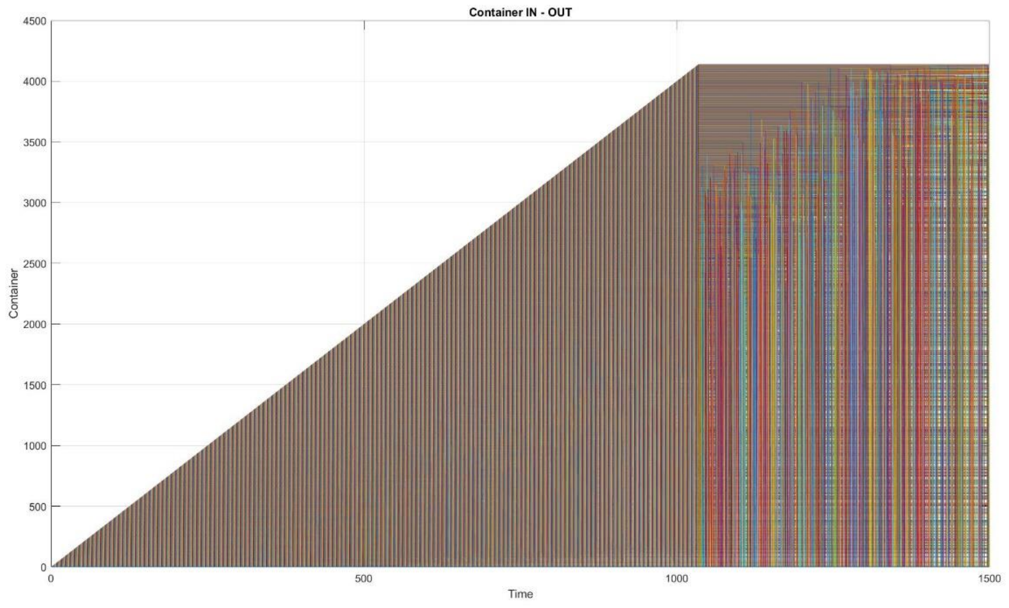


Figure 77 Arrival and exit time of the container in Deliverer agent with 1500 its, 10 ens_max and 4 work max

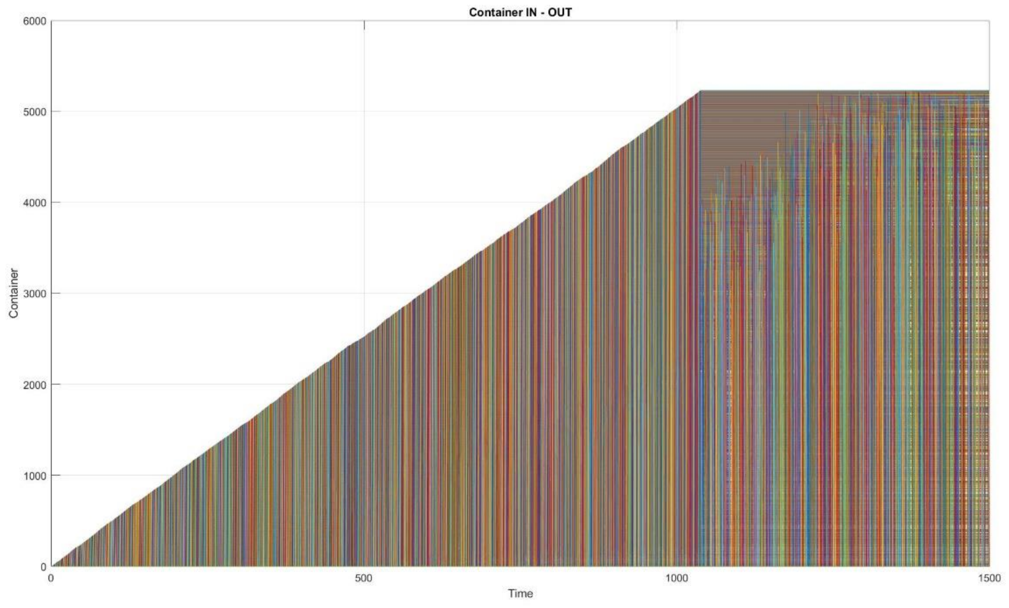


Figure 78 Arrival and exit time of the container in Deliverer agent with 1500 its, 10 ens_max and 6 work max

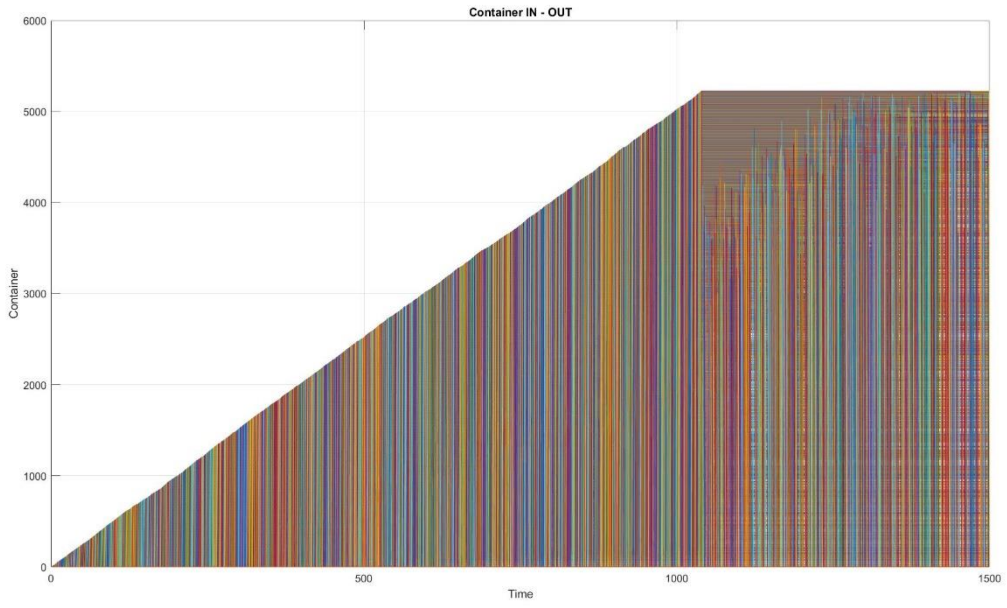


Figure 79 Arrival and exit time of the container in Deliverer agent with 1500 its, 10 ens_max and 8 work max

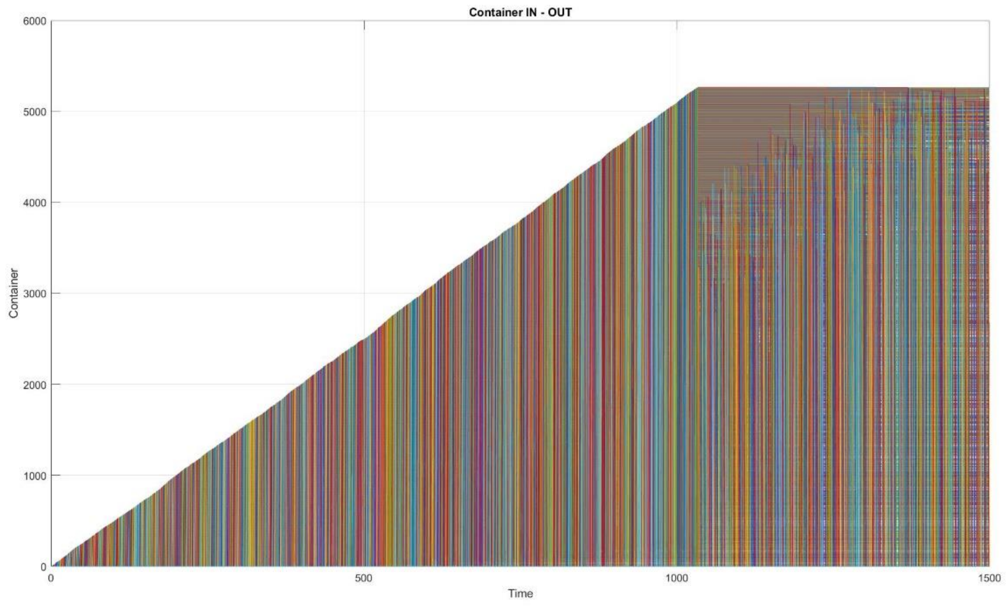


Figure 80 Arrival and exit time of the container in Deliverer agent with 1500 its, 10 ens_max and 10 work max

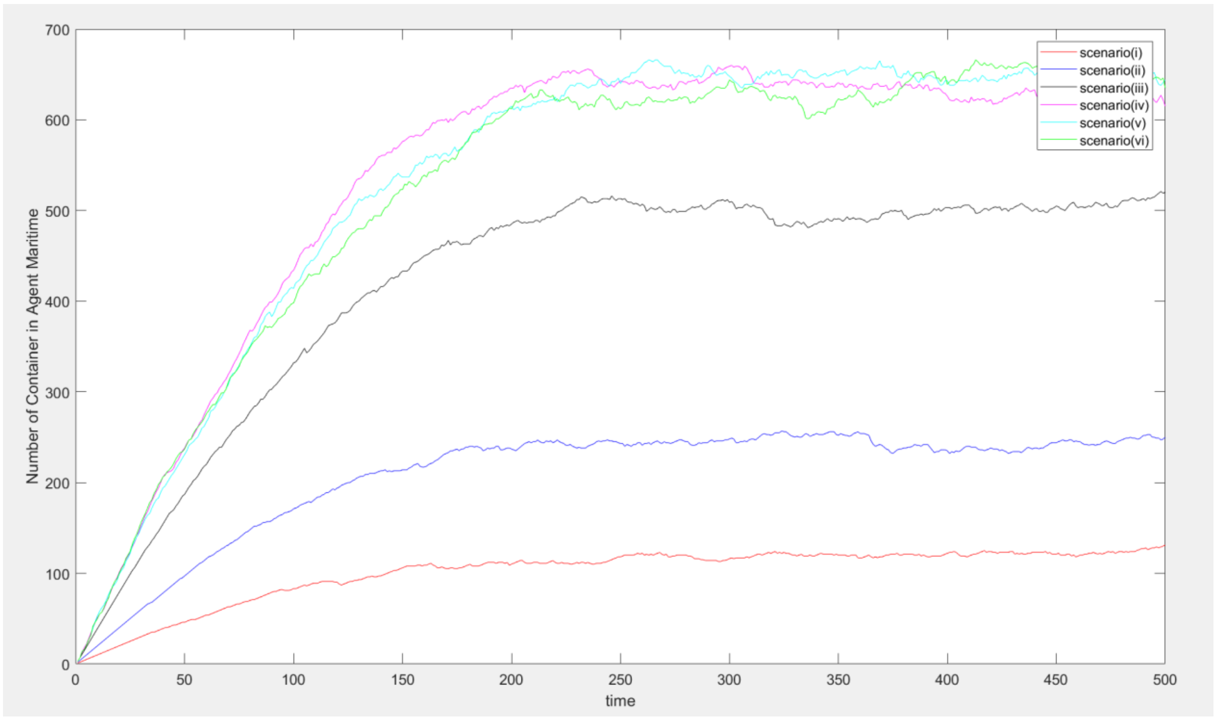


Figure 81 The number of containers in Agent Maritime-500 Iterations

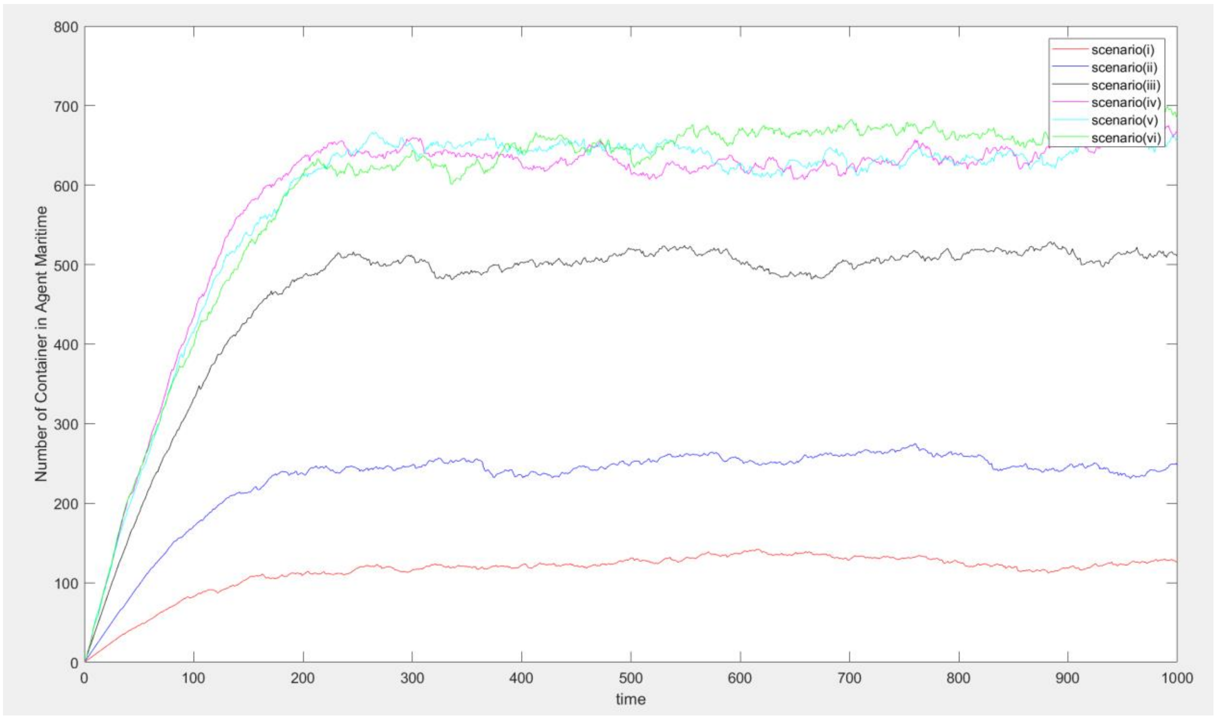


Figure 82 The number of containers in Agent Maritime-1000 Iterations

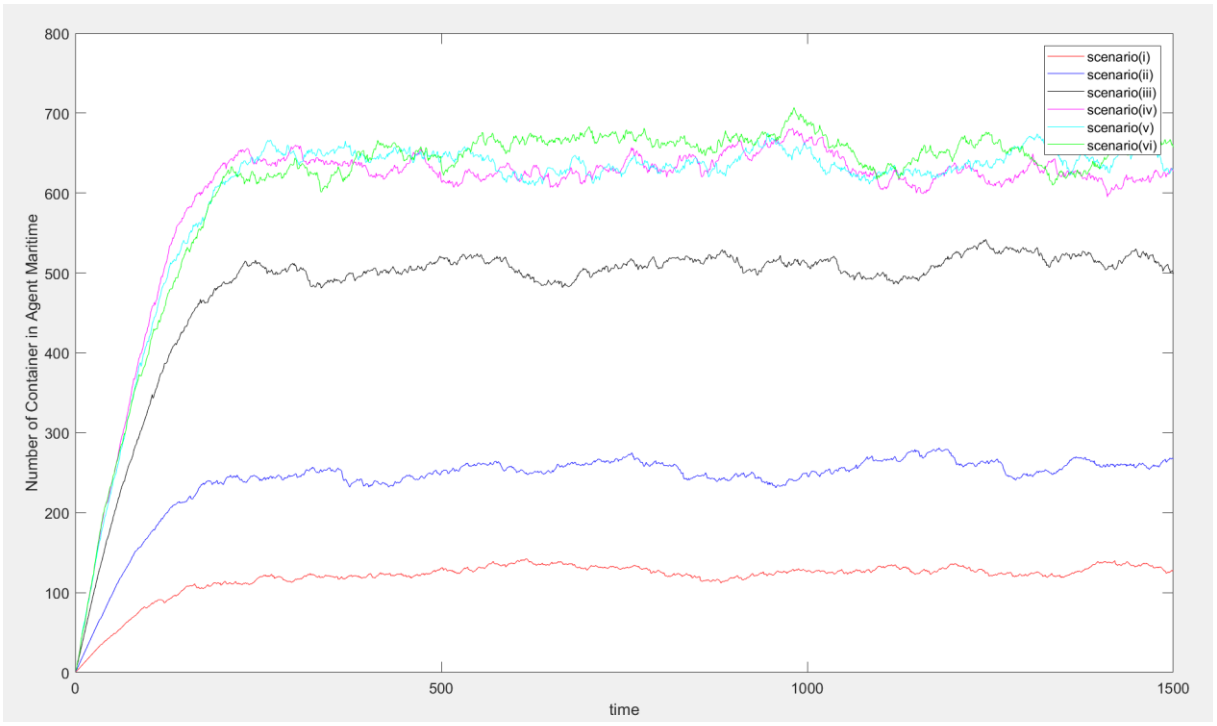


Figure 83 The number of containers in Agent Maritime-1500 Iterations

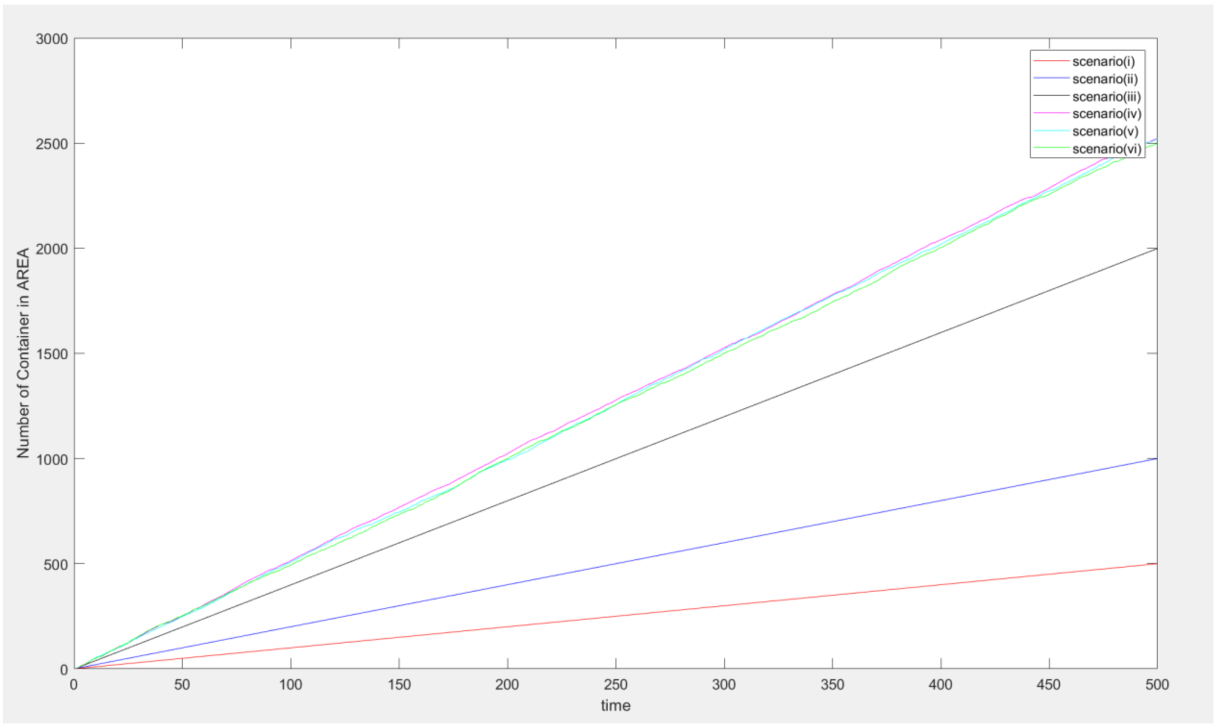


Figure 84 The number of containers in Agent Area-500 Iterations

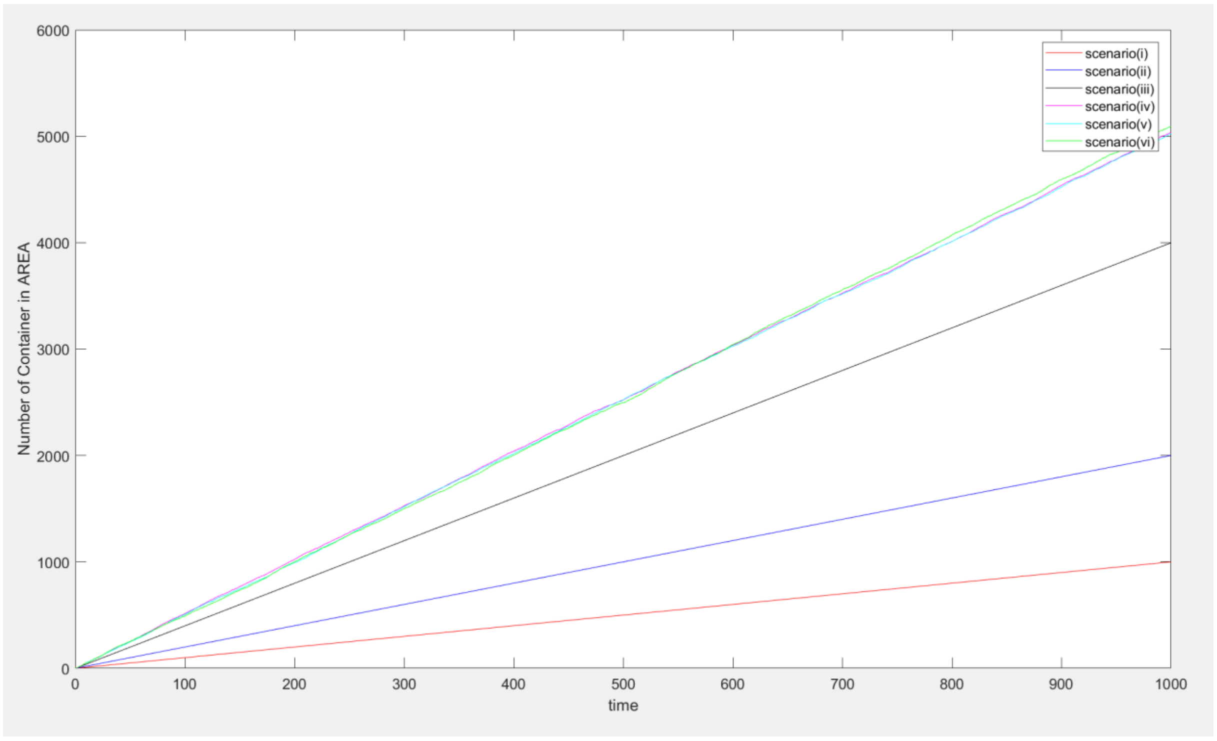


Figure 85 The number of containers in Agent Area-1000 Iterations

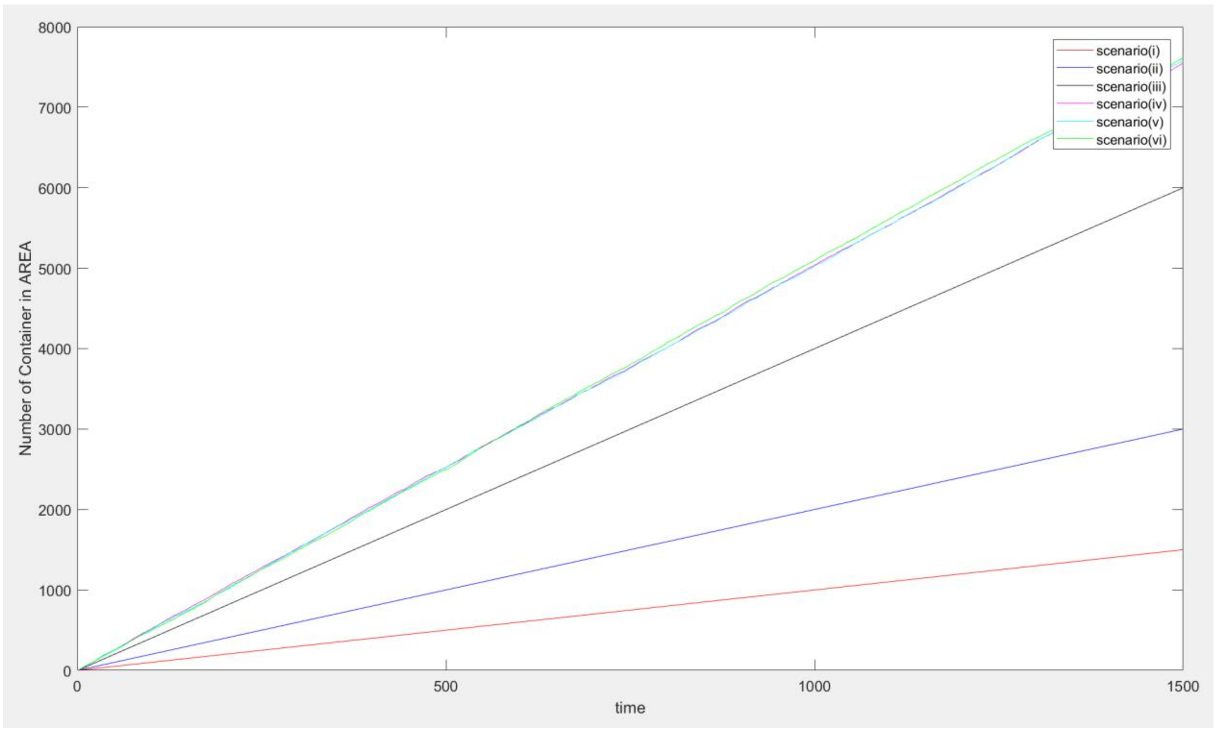


Figure 86 The number of containers in Agent Area-1500 Iterations

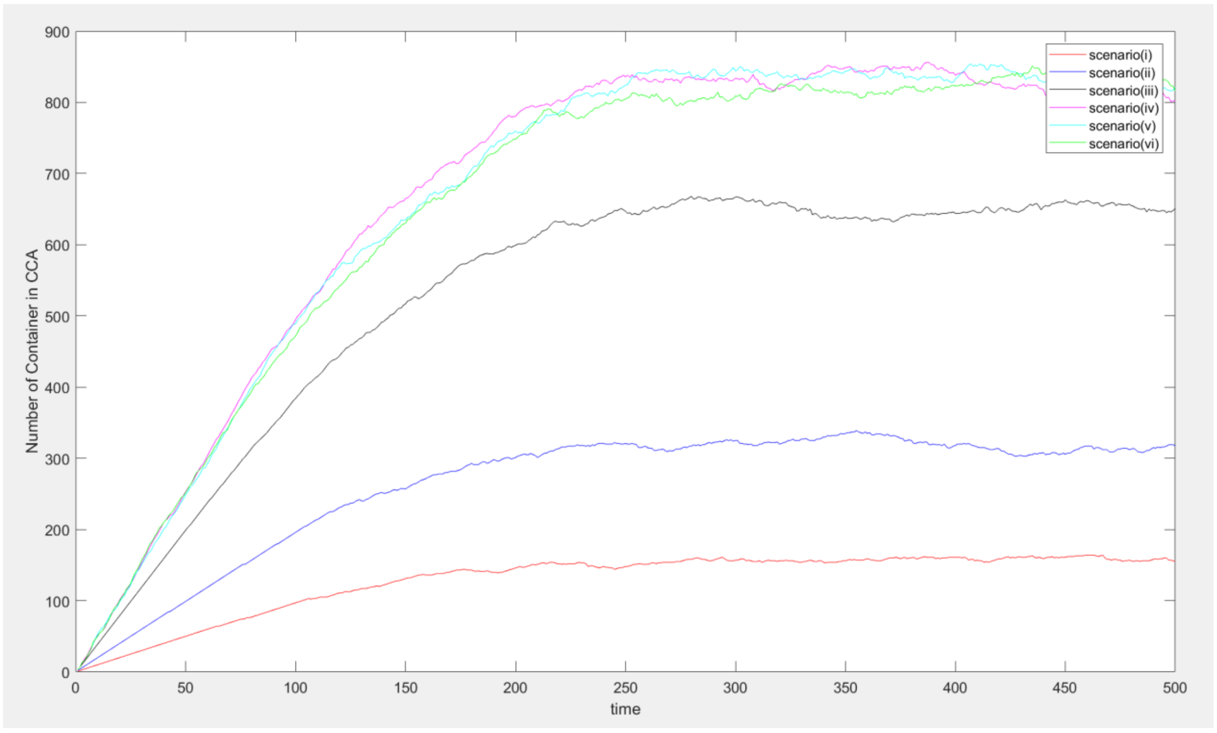


Figure 87 The number of containers in Agent CCA-500 Iterations

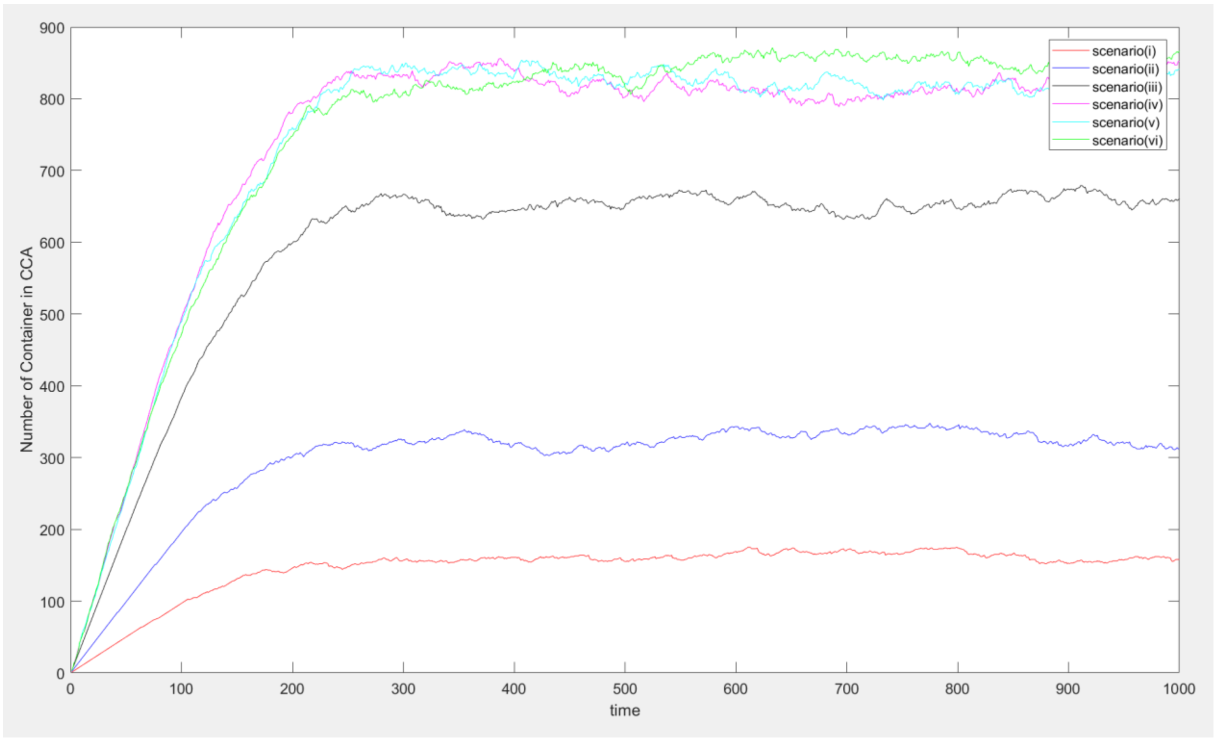


Figure 88 The number of containers in Agent CCA-1000 Iterations

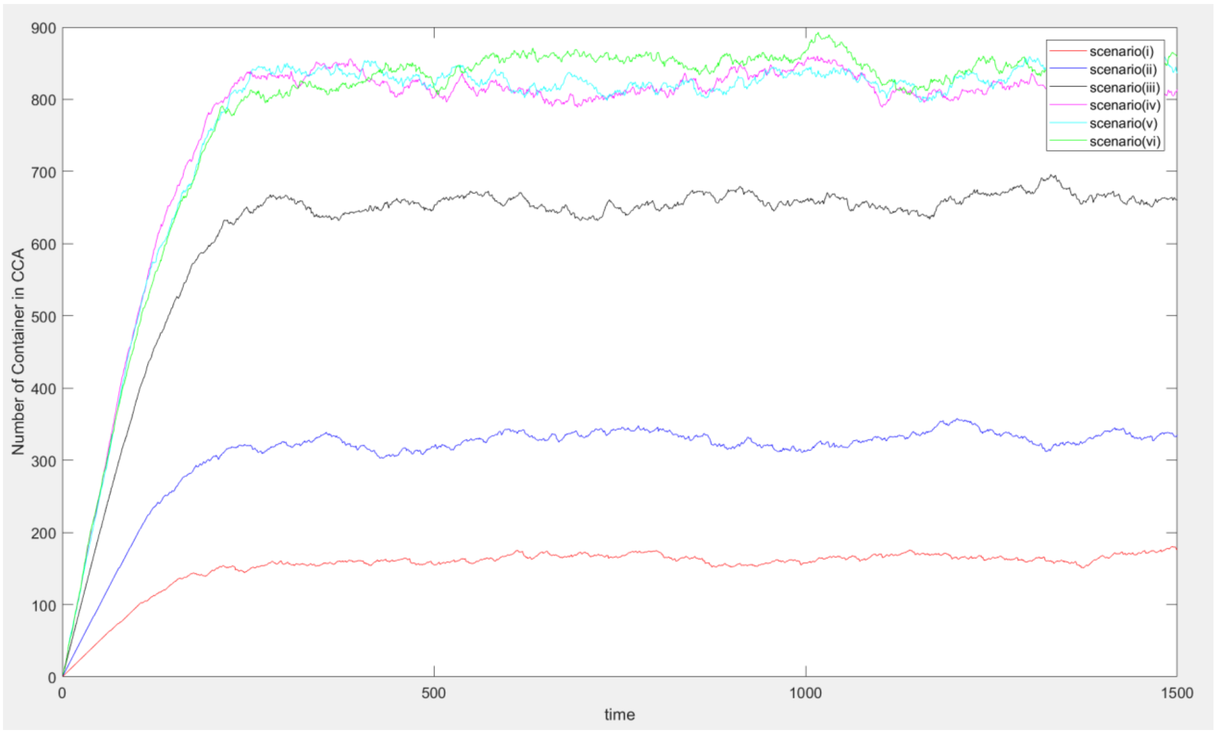


Figure 89 The number of containers in Agent CCA-1500 Iterations

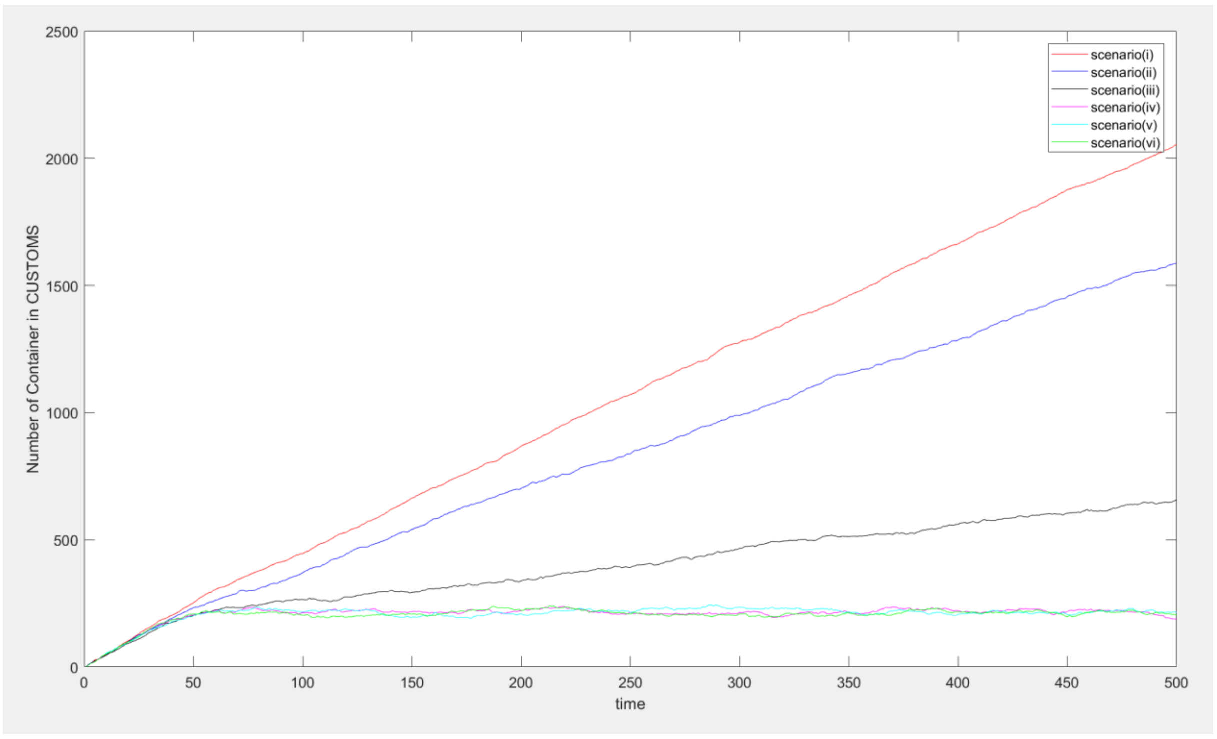


Figure 90 The number of containers in Agent Customs-500 Iterations

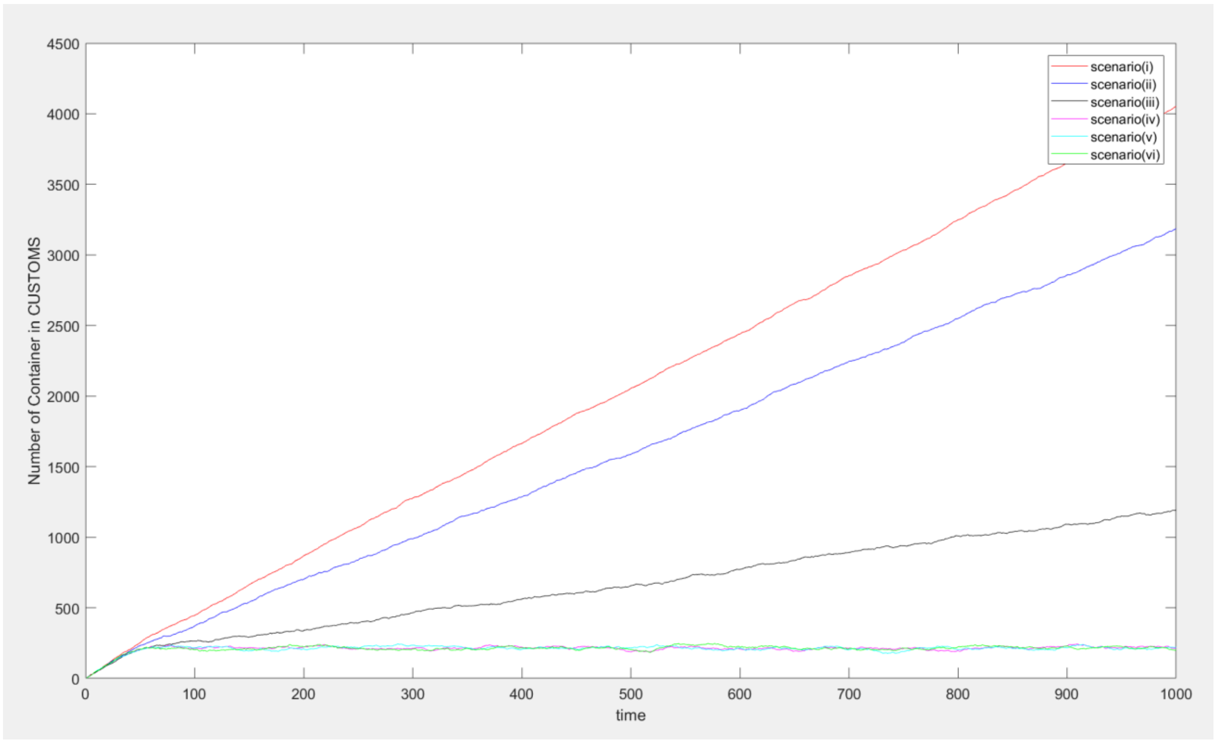


Figure 91 The number of containers in Agent Customs-1000 Iterations

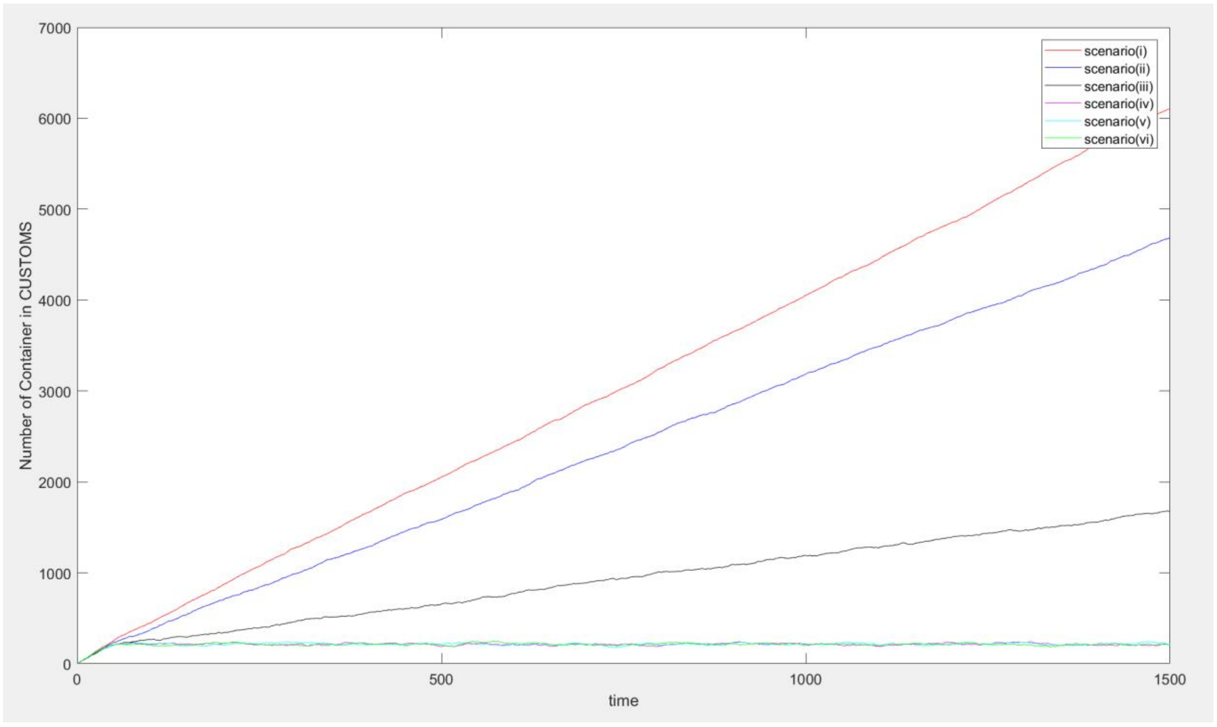


Figure 92 The number of containers in Agent Customs-1500 Iterations

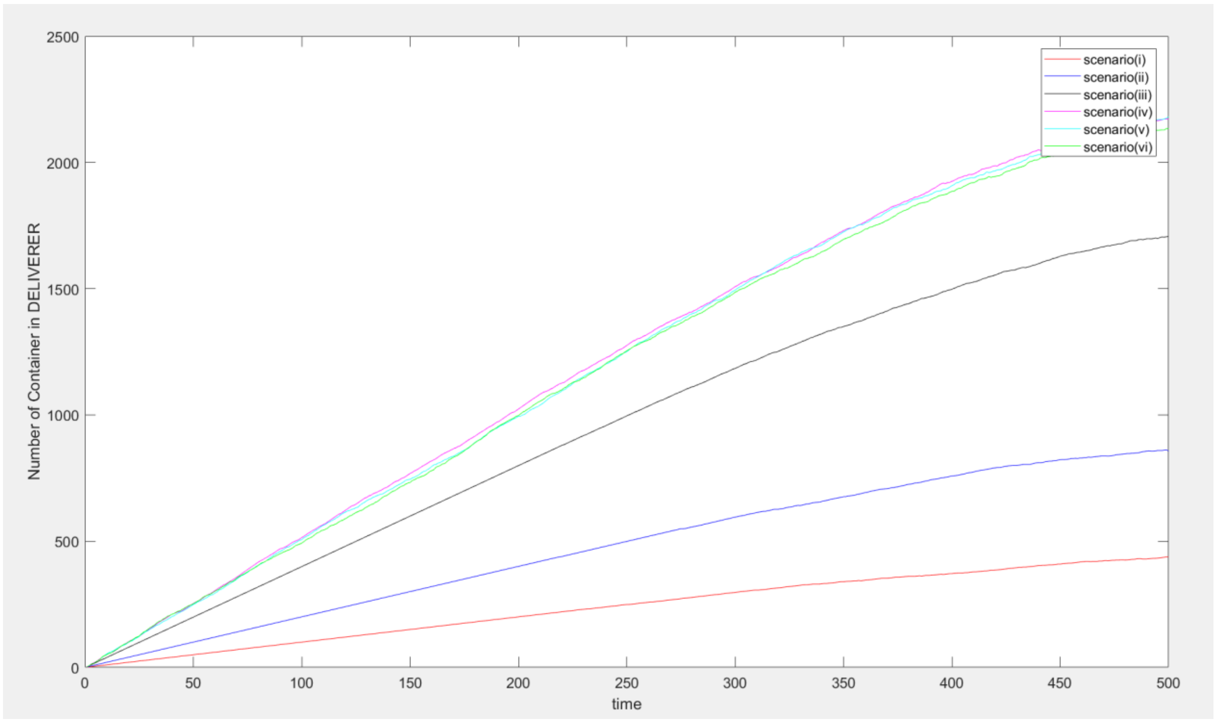


Figure 93 The number of containers in Agent Deliverer-500 Iterations

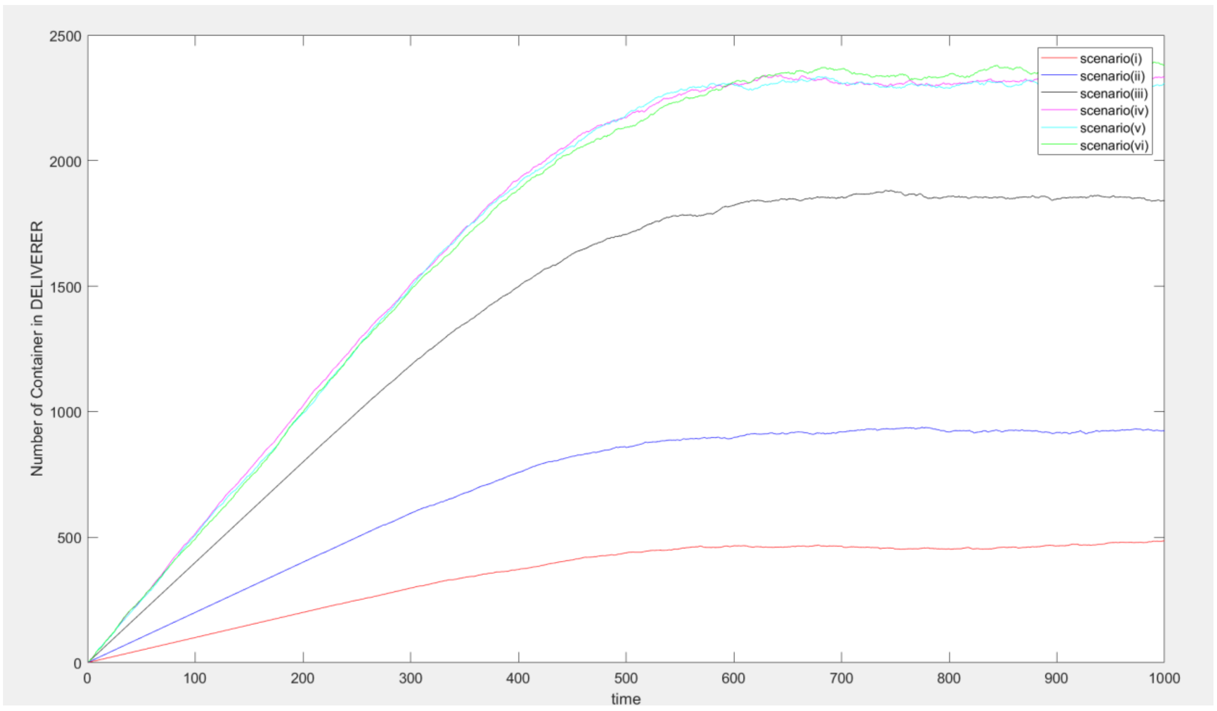


Figure 94 The number of containers in Agent Deliverer-1000 Iterations

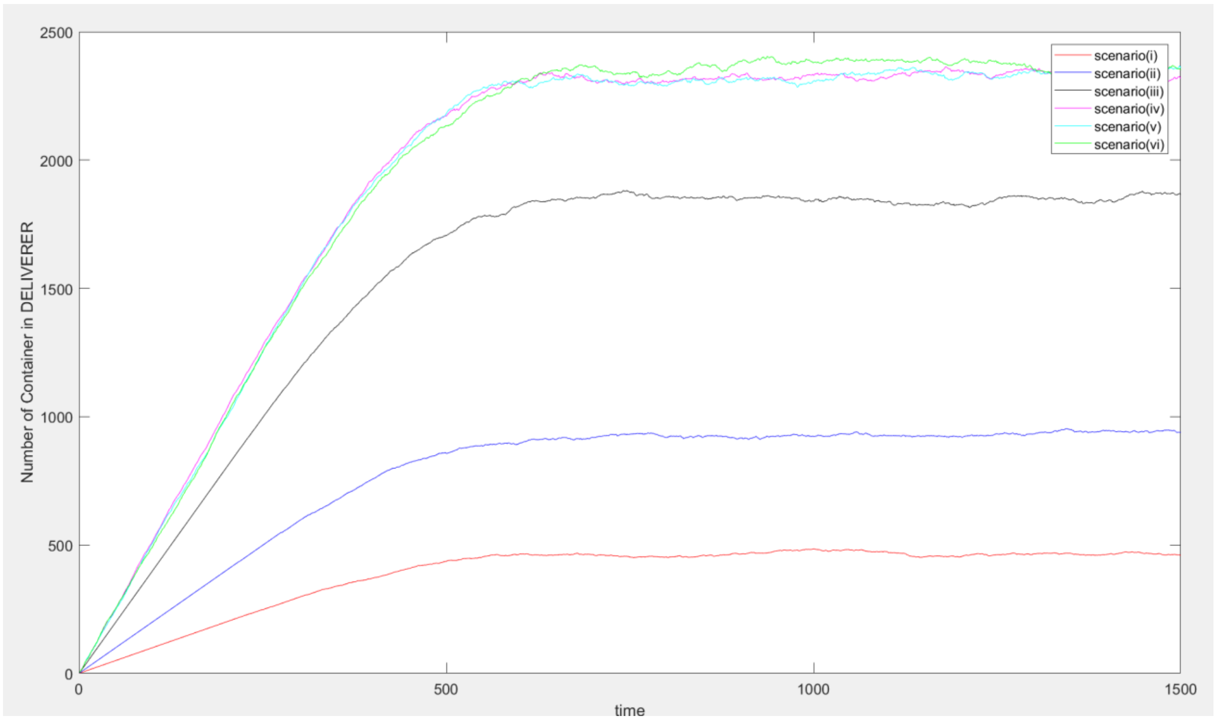


Figure 95 The number of containers in Agent Deliverer-1500 Iterations

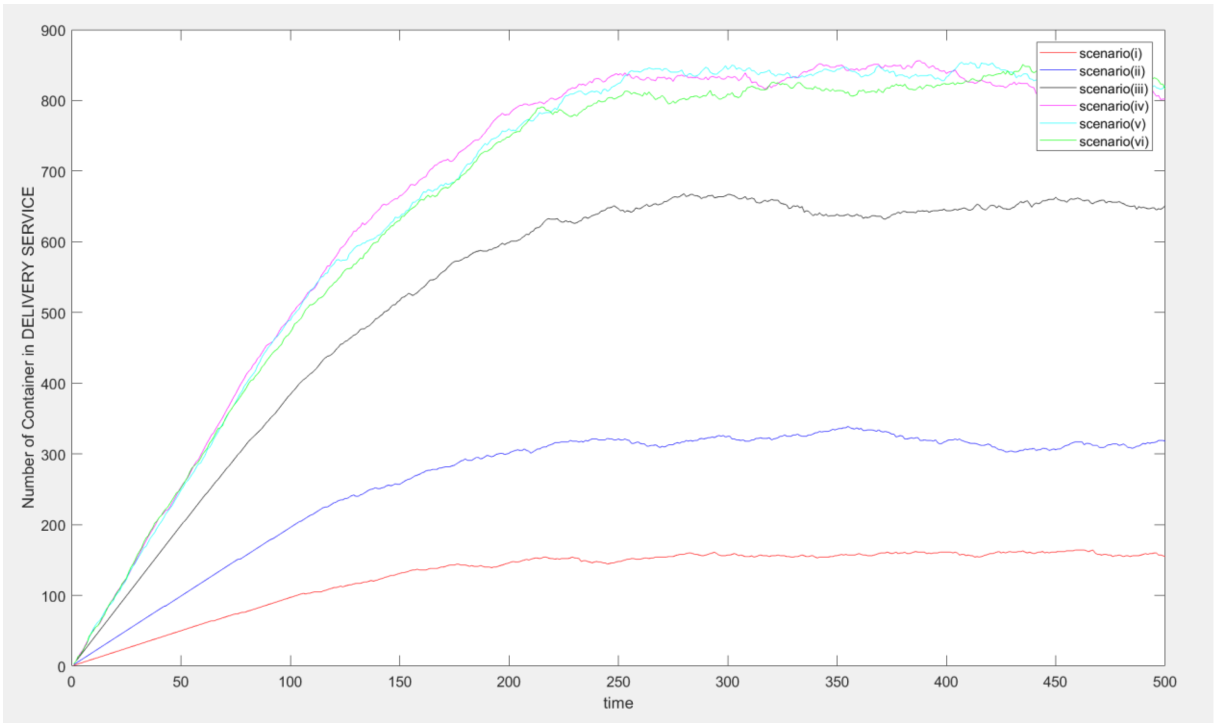


Figure 96 The number of containers in Agent Delivery Service-500 Iterations

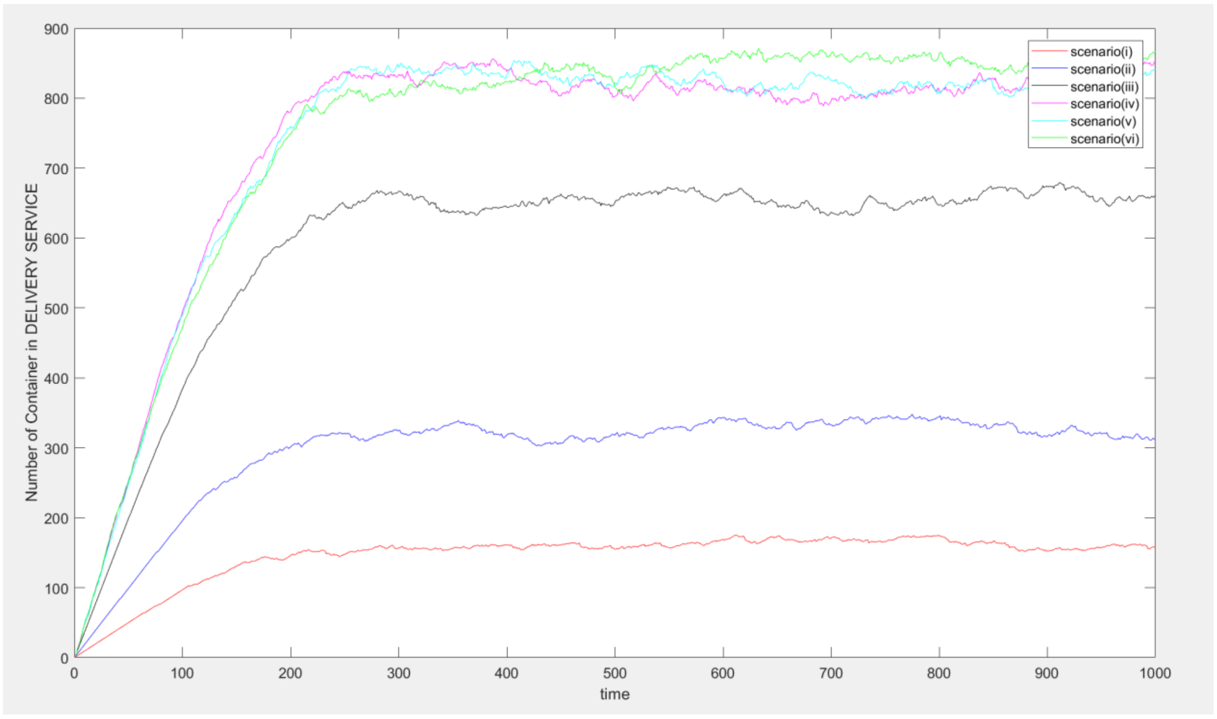


Figure 97 The number of containers in Agent Delivery Service-1000 Iterations

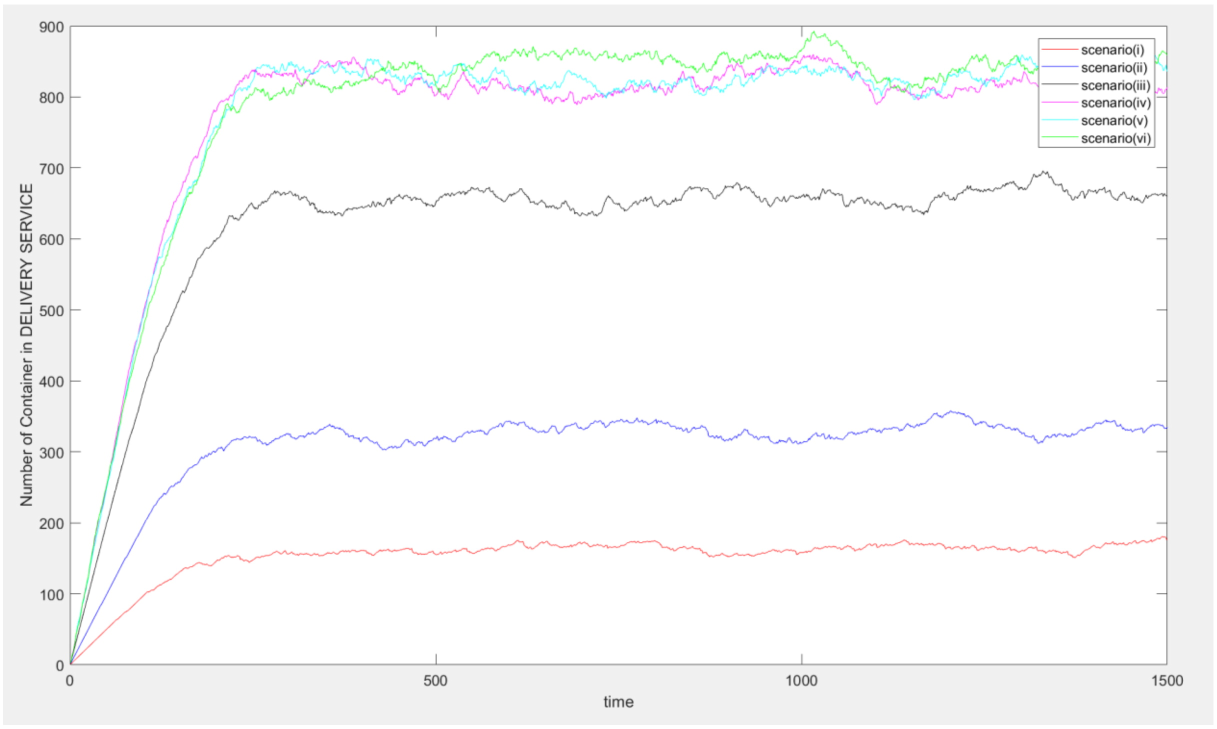


Figure 98 The number of containers in Agent Delivery Service-1500 Iterations

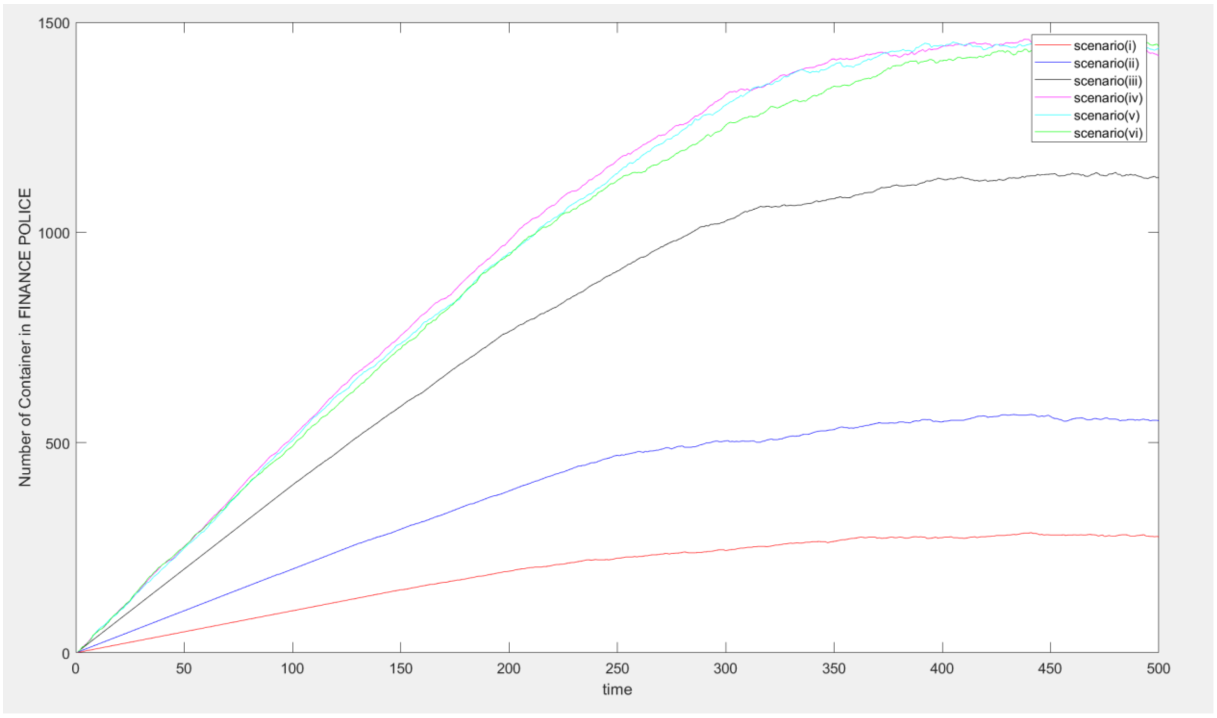


Figure 99 The number of containers in Agent Finance Police-500 Iterations

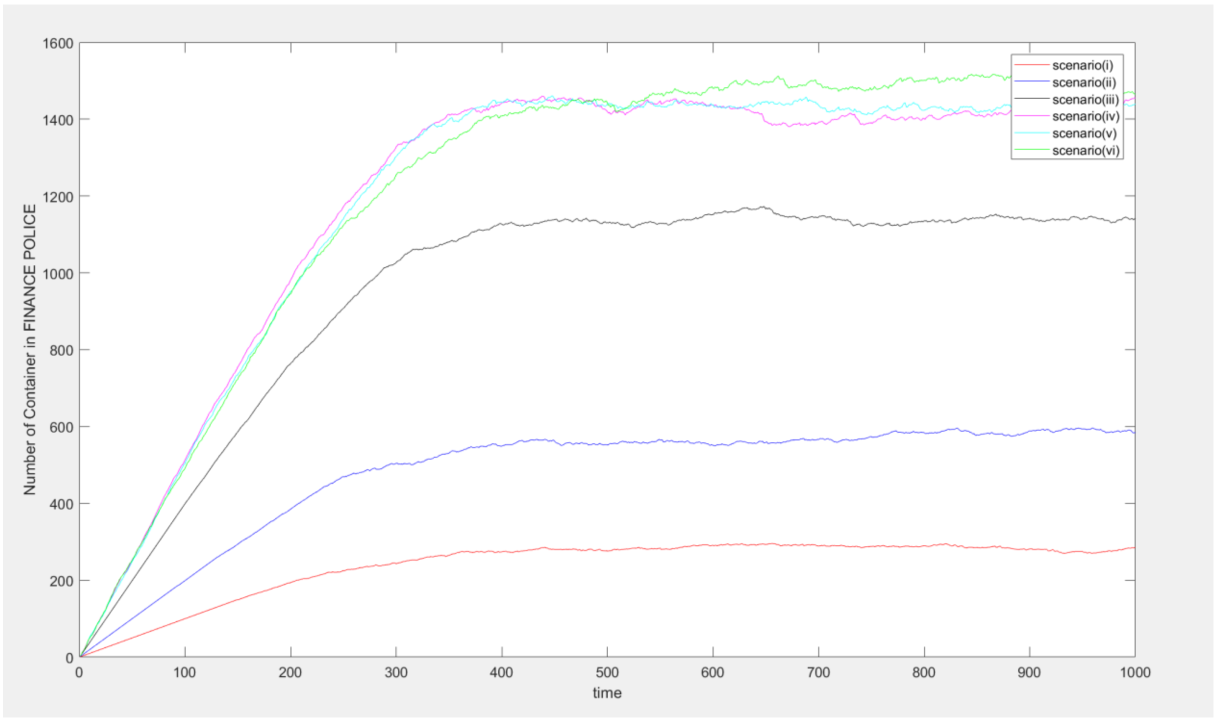


Figure 100 The number of containers in Agent Finance Police-1000 Iterations

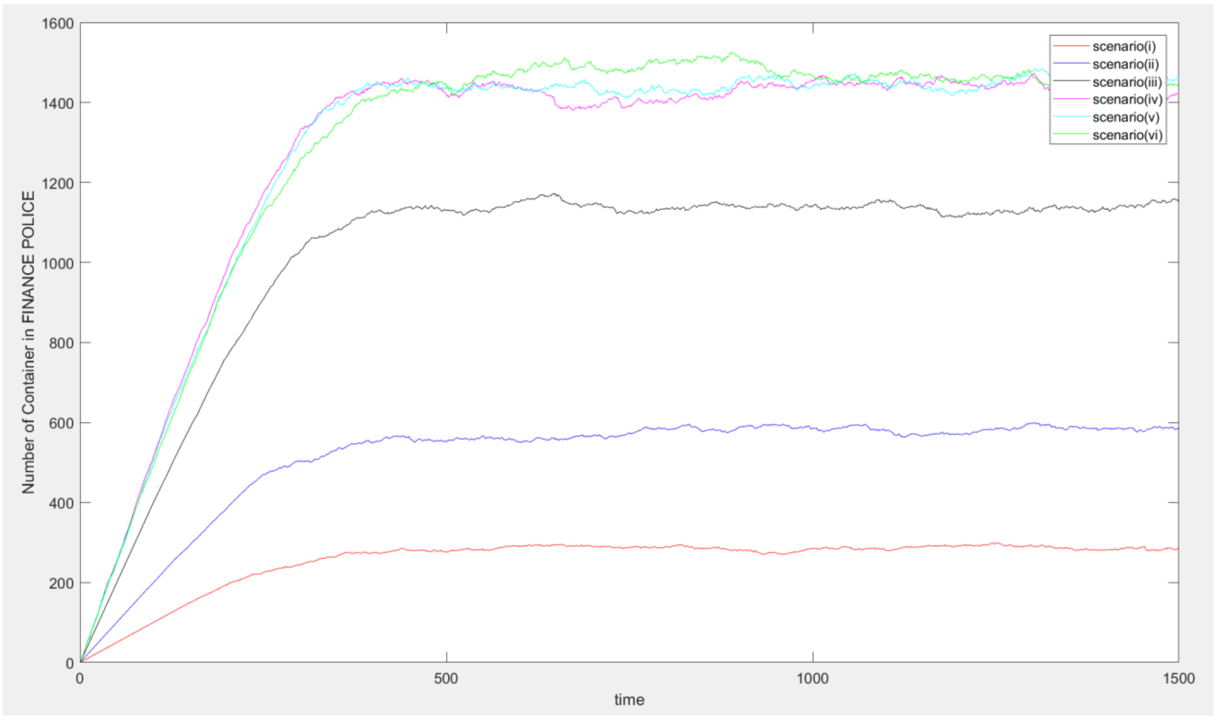


Figure 101 The number of containers in Agent Finance Police-1500 Iterations

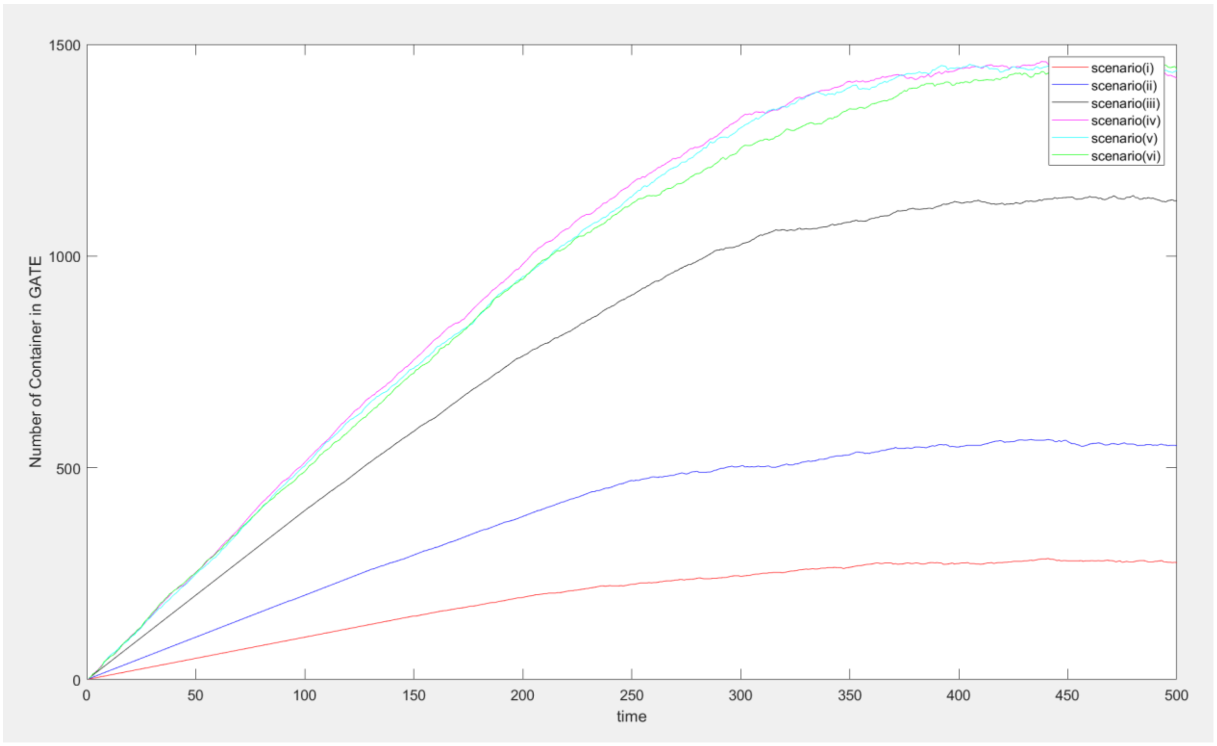


Figure 102 The number of containers in Agent Gate -500 Iterations

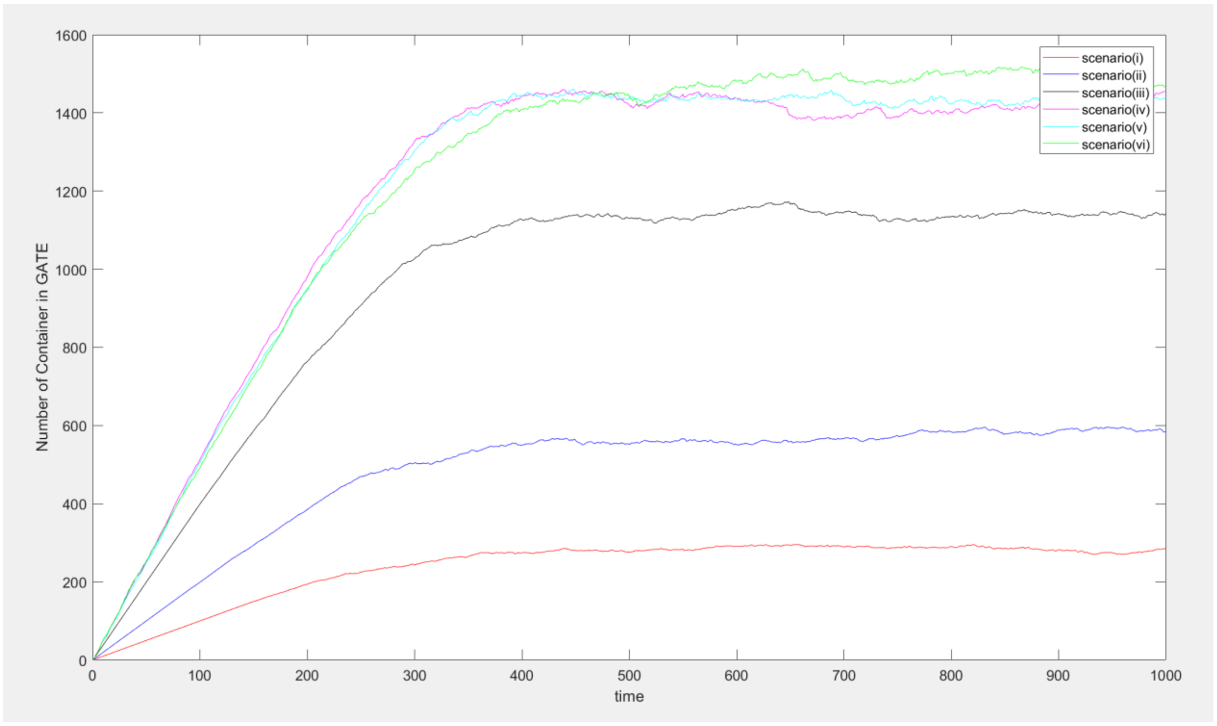


Figure 103 The number of containers in Agent Gate -1000 Iterations

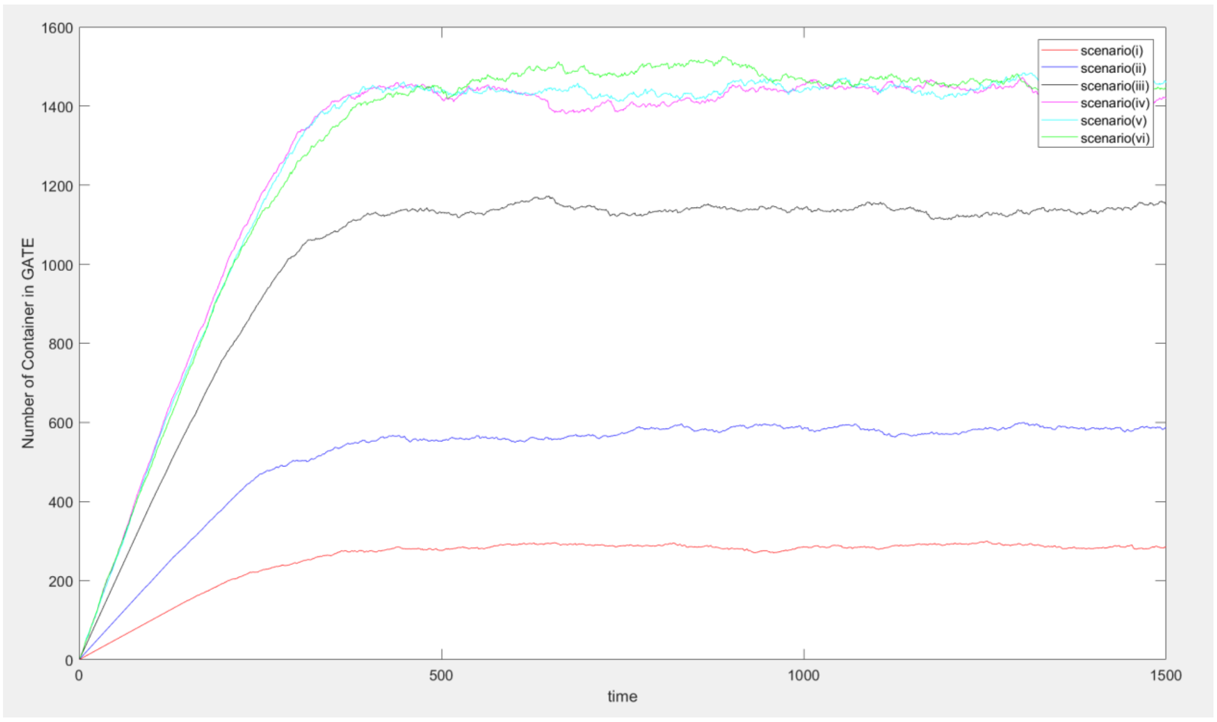


Figure 104 The number of containers in Agent Gate -1500 Iterations

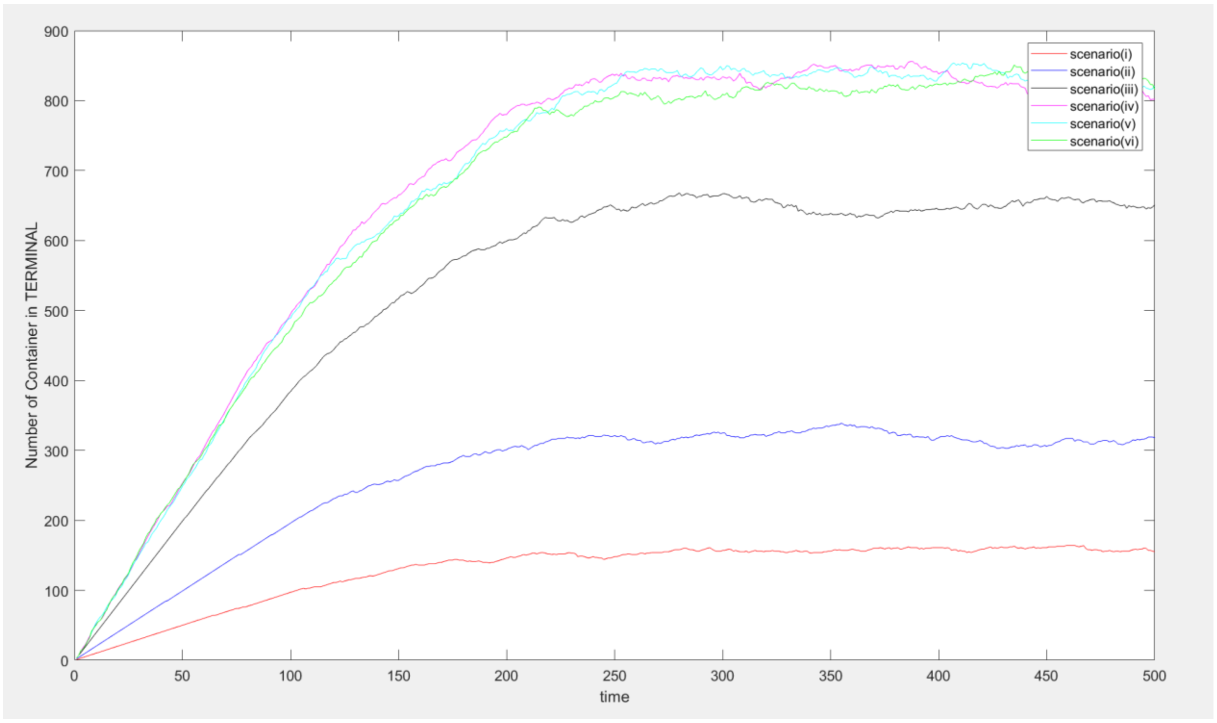


Figure 105 The number of containers in Agent Terminal -500 Iterations

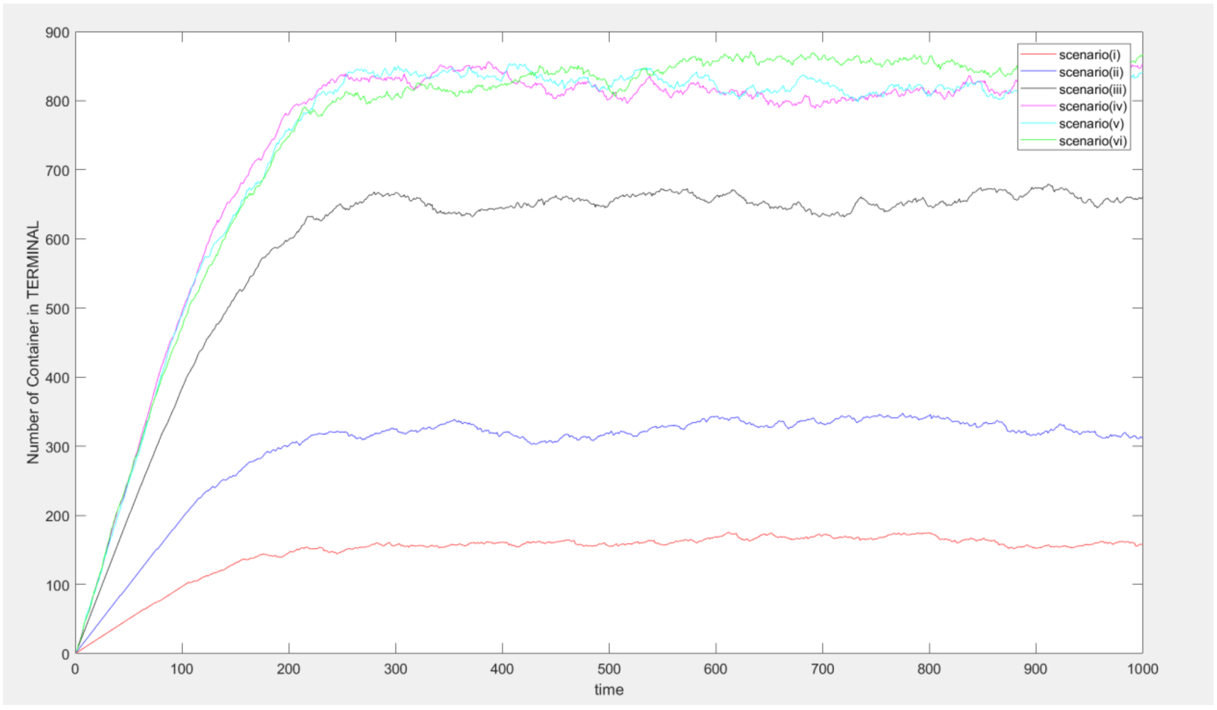


Figure 106 The number of containers in Agent Terminal -1000 Iterations

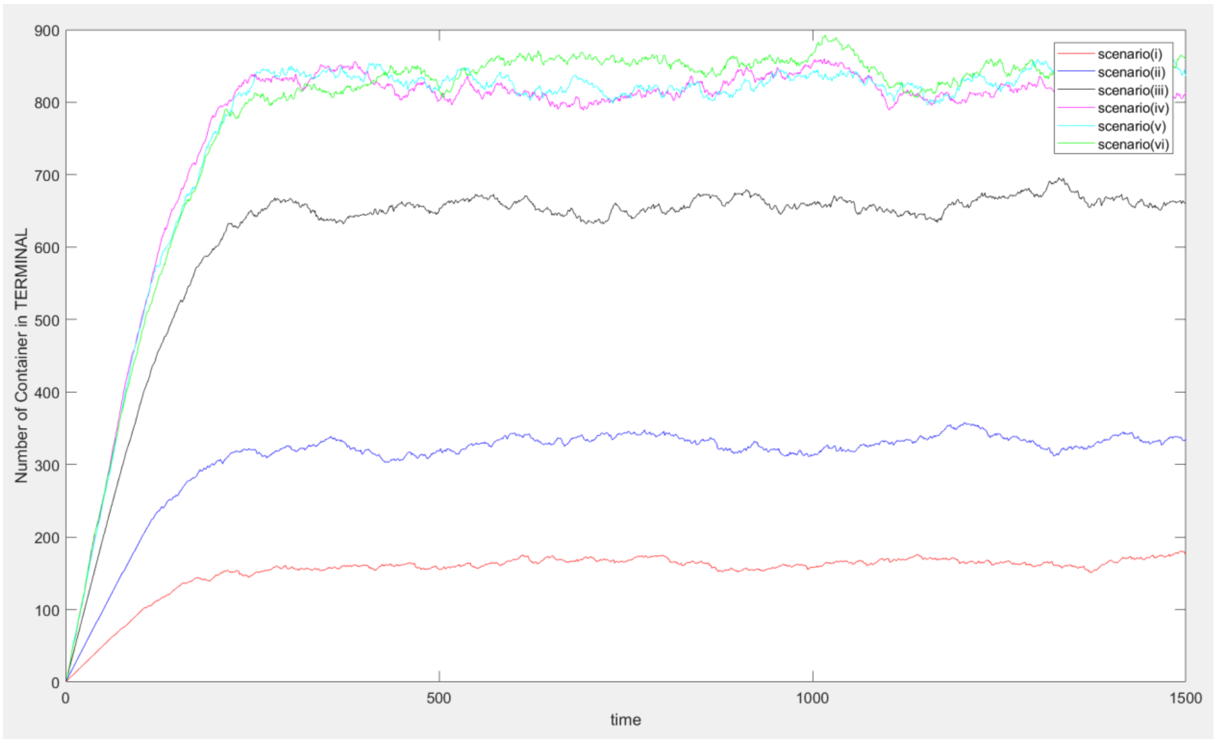


Figure 107 The number of containers in Agent Terminal -1500 Iterations

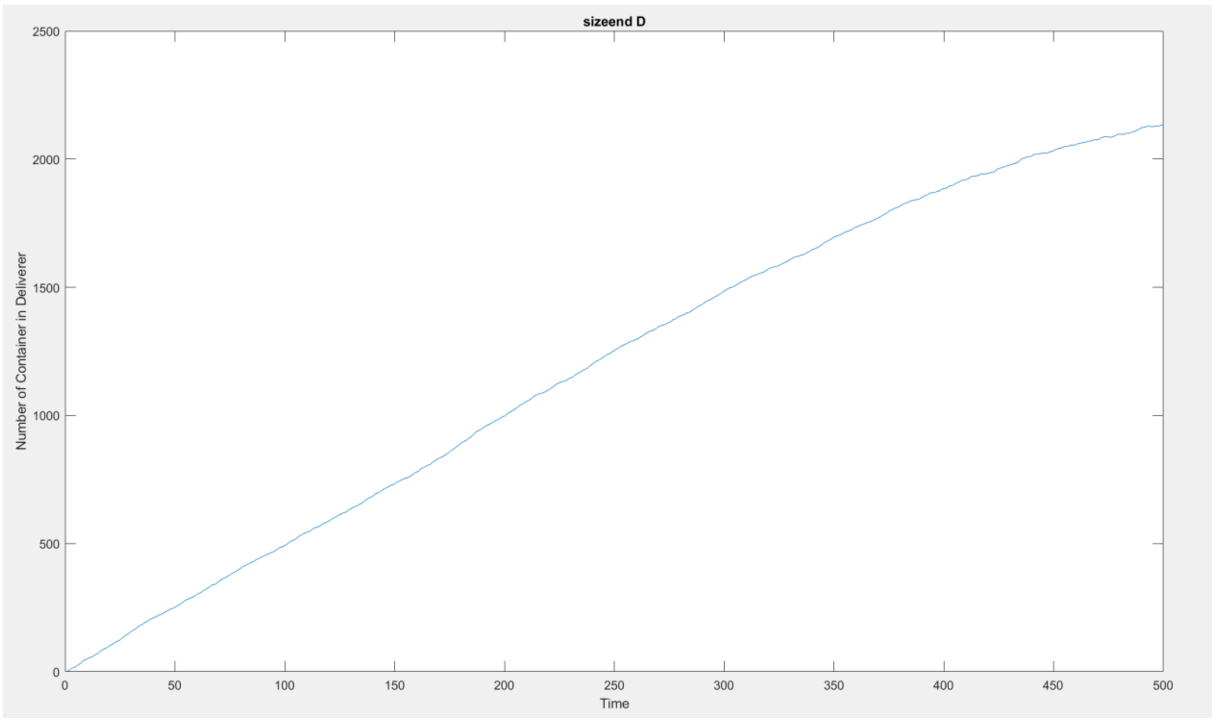


Figure 108 Sizeend of the Deliverer Agent- 500 its ens_max 10 and work max 10

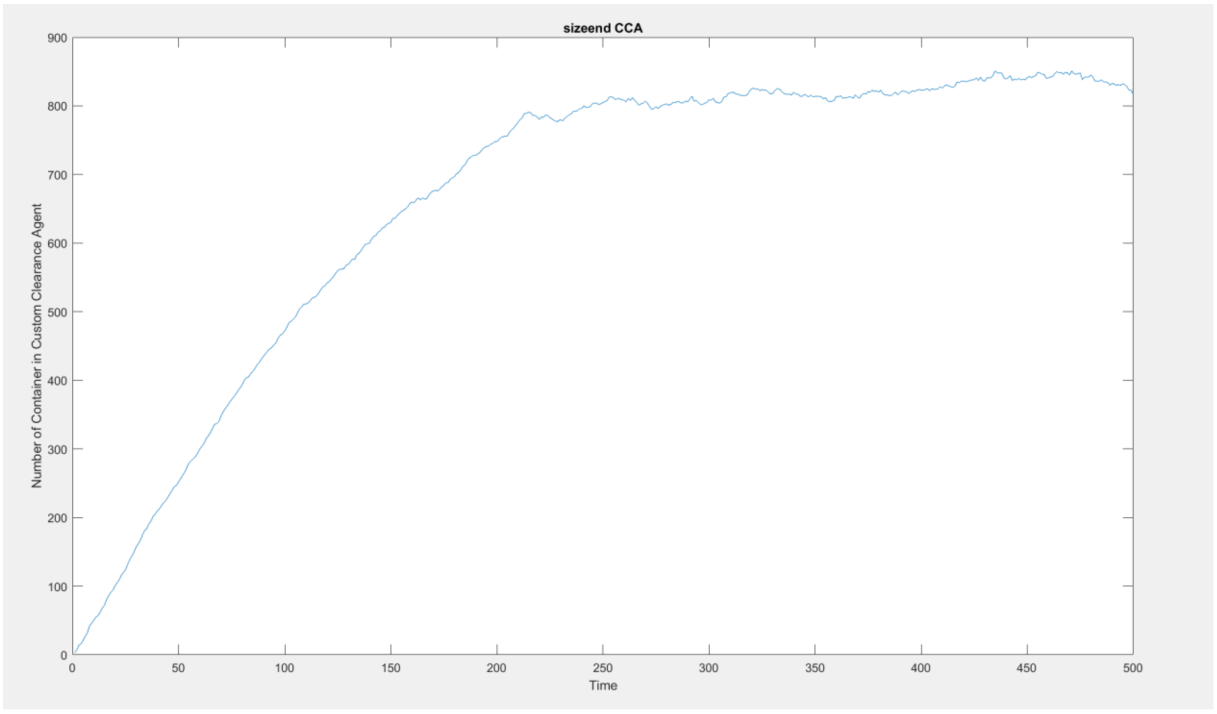


Figure 109 Sizeend of the CCA Agent- 500 its ens_max 10 and work max 10

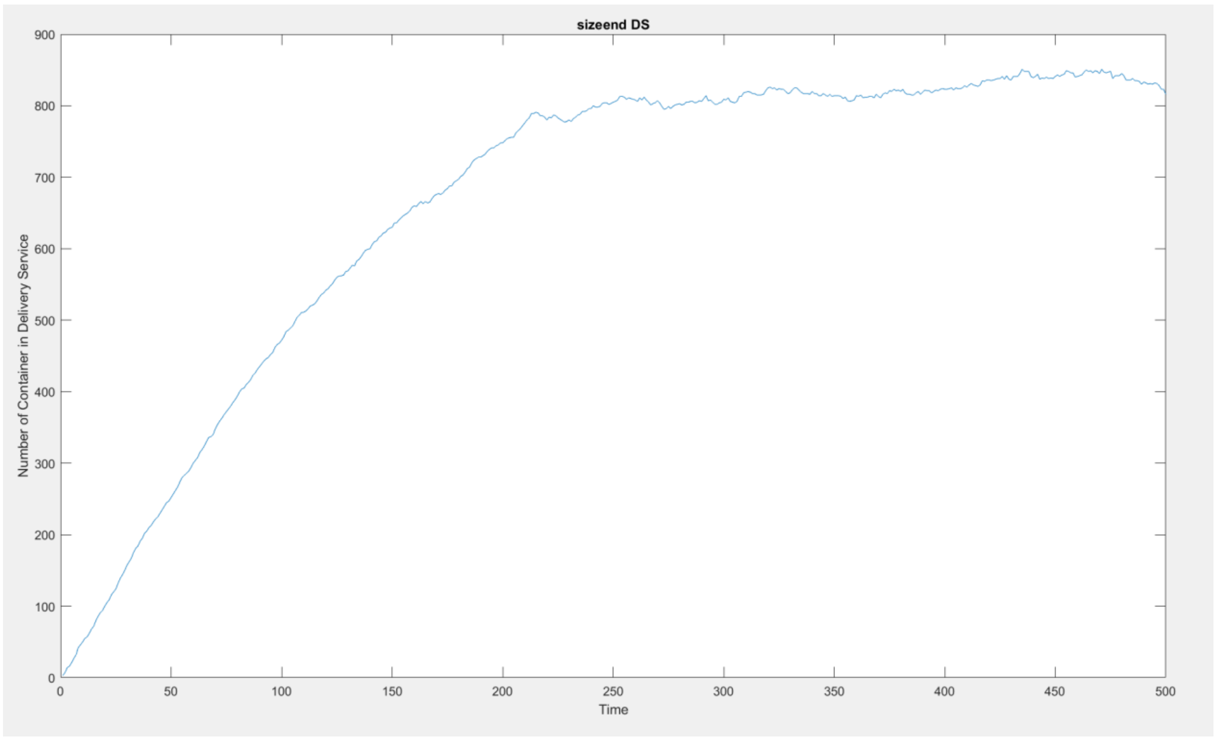


Figure 110 Sizeend of the Delivery Service Agent- 500 its ens_max 10 and work max 10

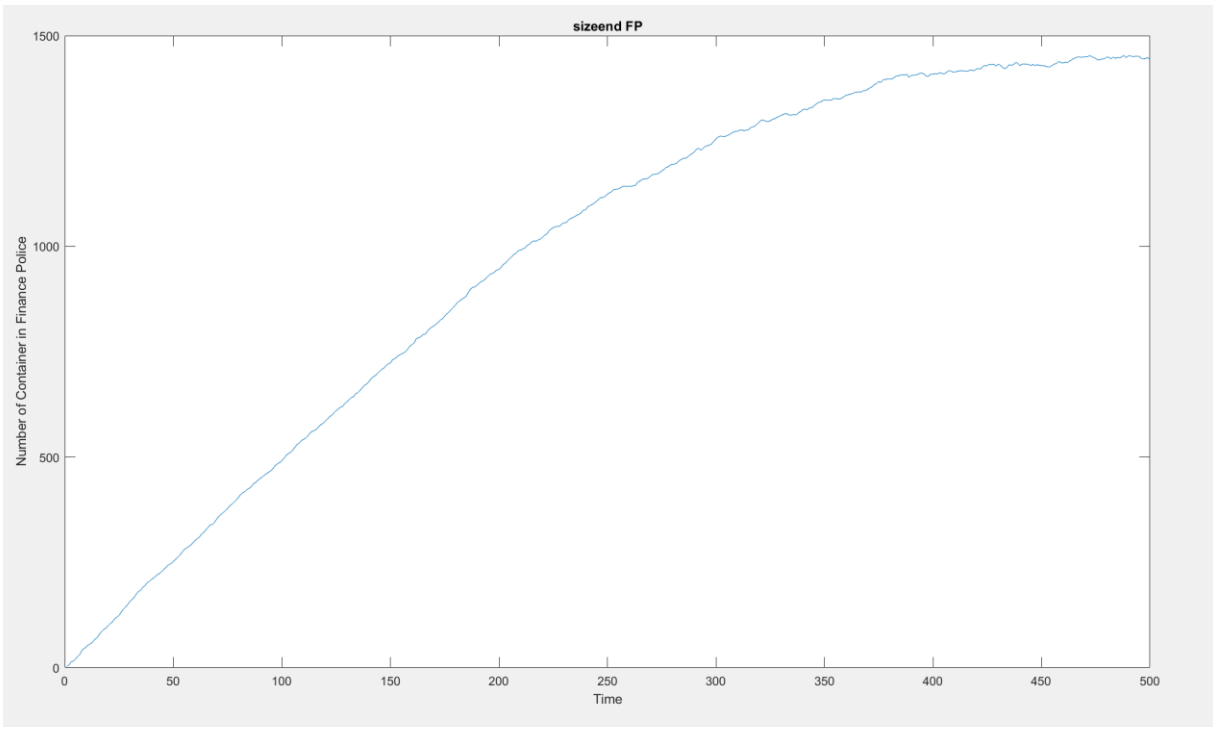


Figure 111 Sizeend of the Finance Police Agent- 500 its ens_max 10 and work max 10

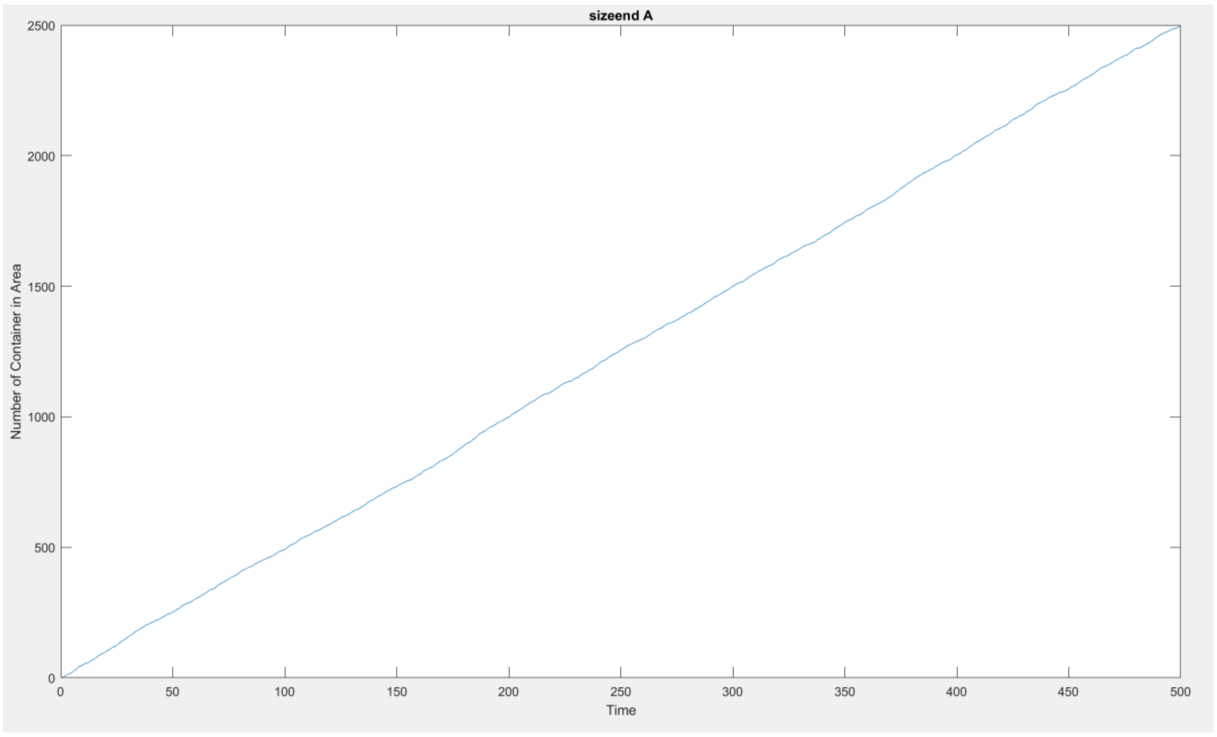


Figure 112 Sizeend of the Area Agent- 500 its ens_max 10 and work max 10

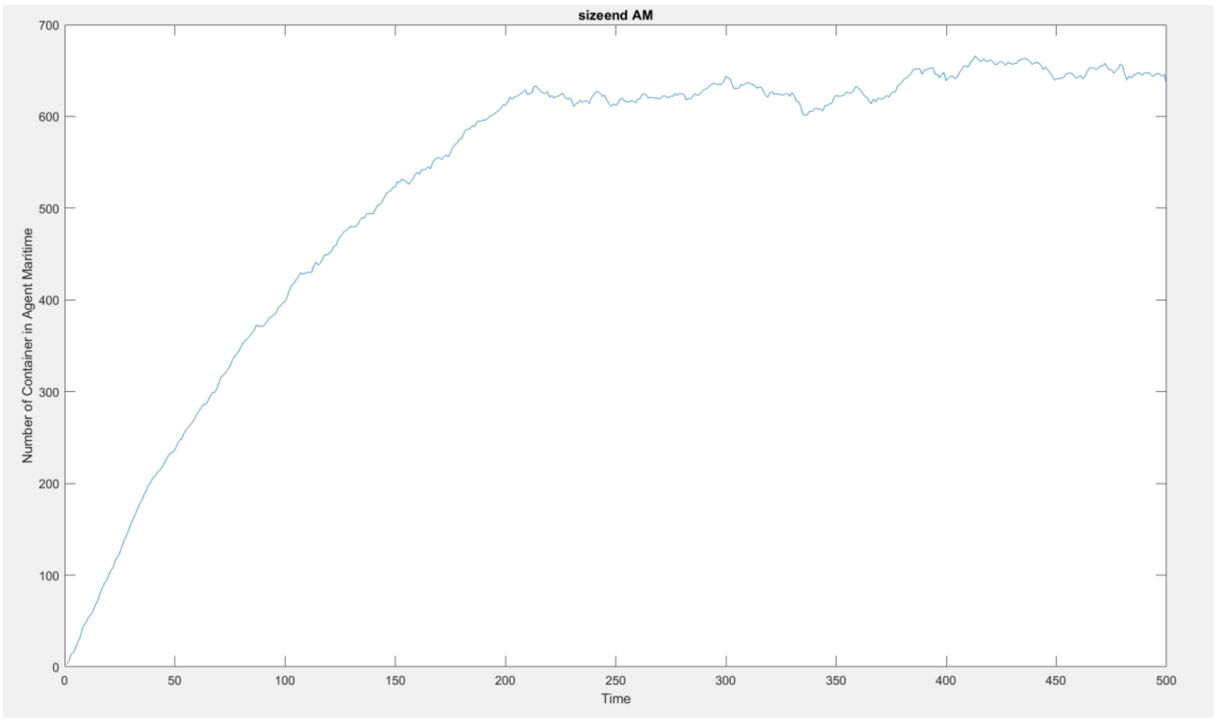


Figure 113 Sizeend of the Agent Maritime- 500 its ens_max 10 and work max 10

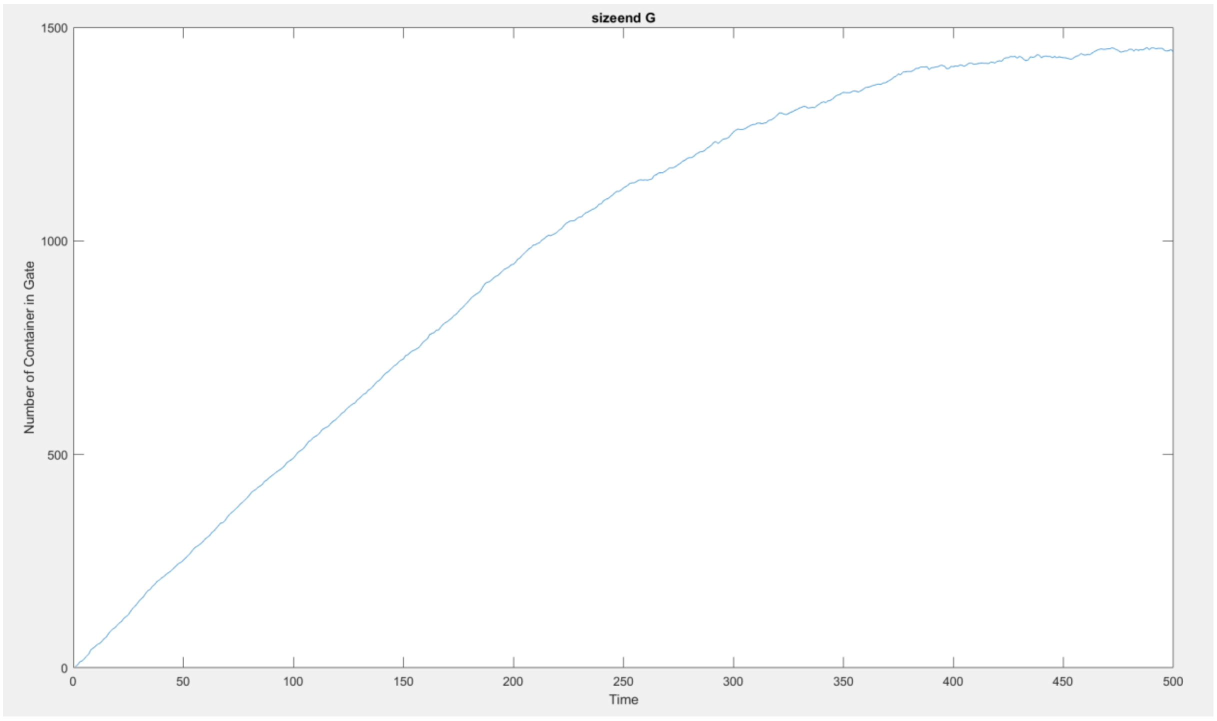


Figure 114 Sizeend of the Gate Agent - 500 its ens_max 10 and work max 10

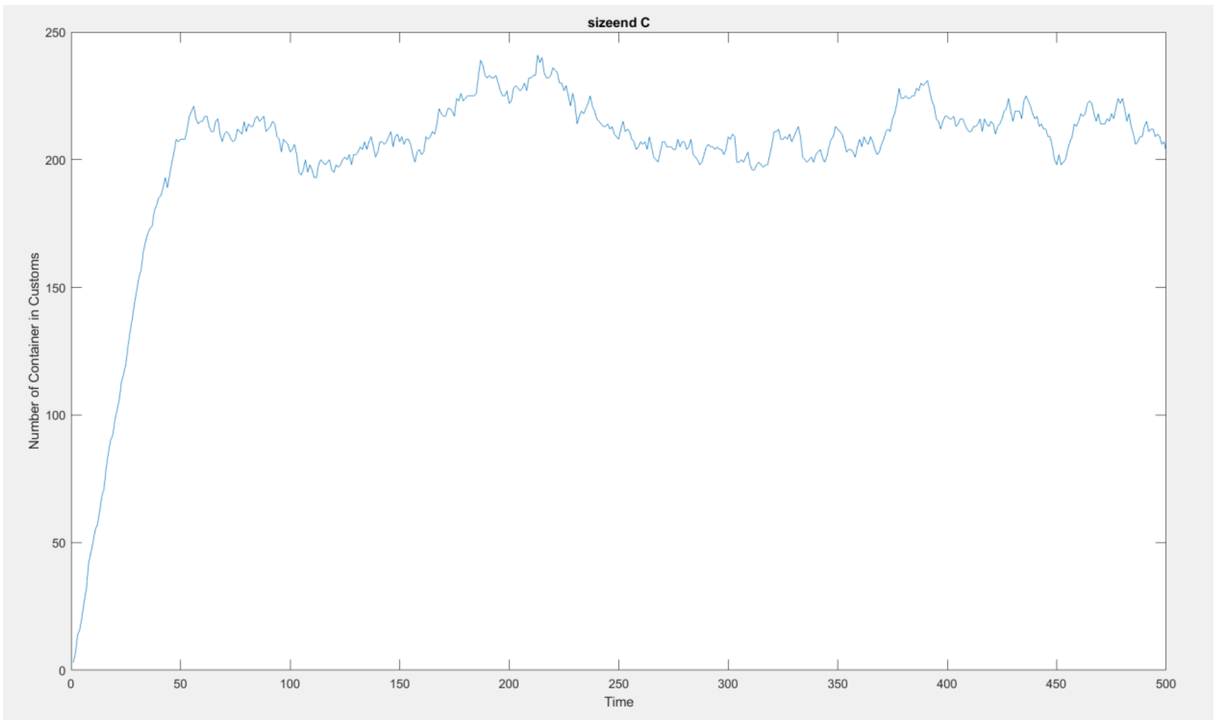


Figure 115 Sizeend of the Customs Agent - 500 its ens_max 10 and work max 10

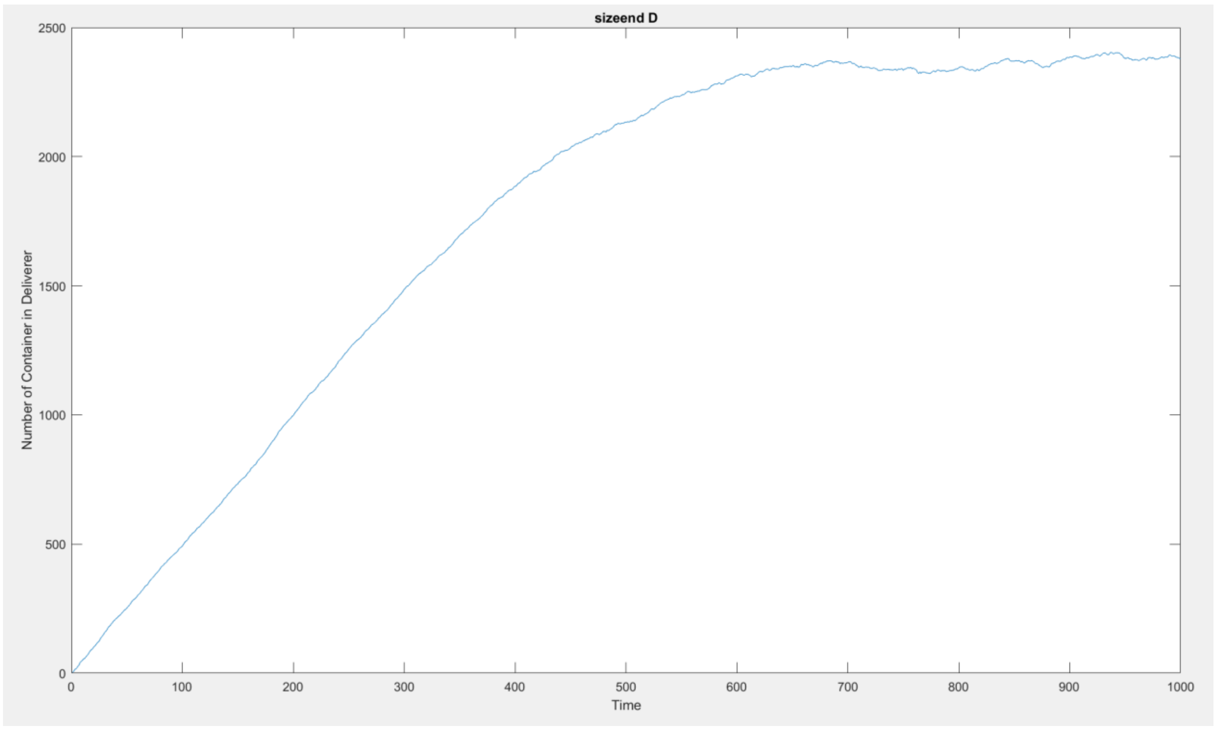


Figure 116 Sizeend of the Deliverer Agent - 1000 its ens_max 10 and work max 10

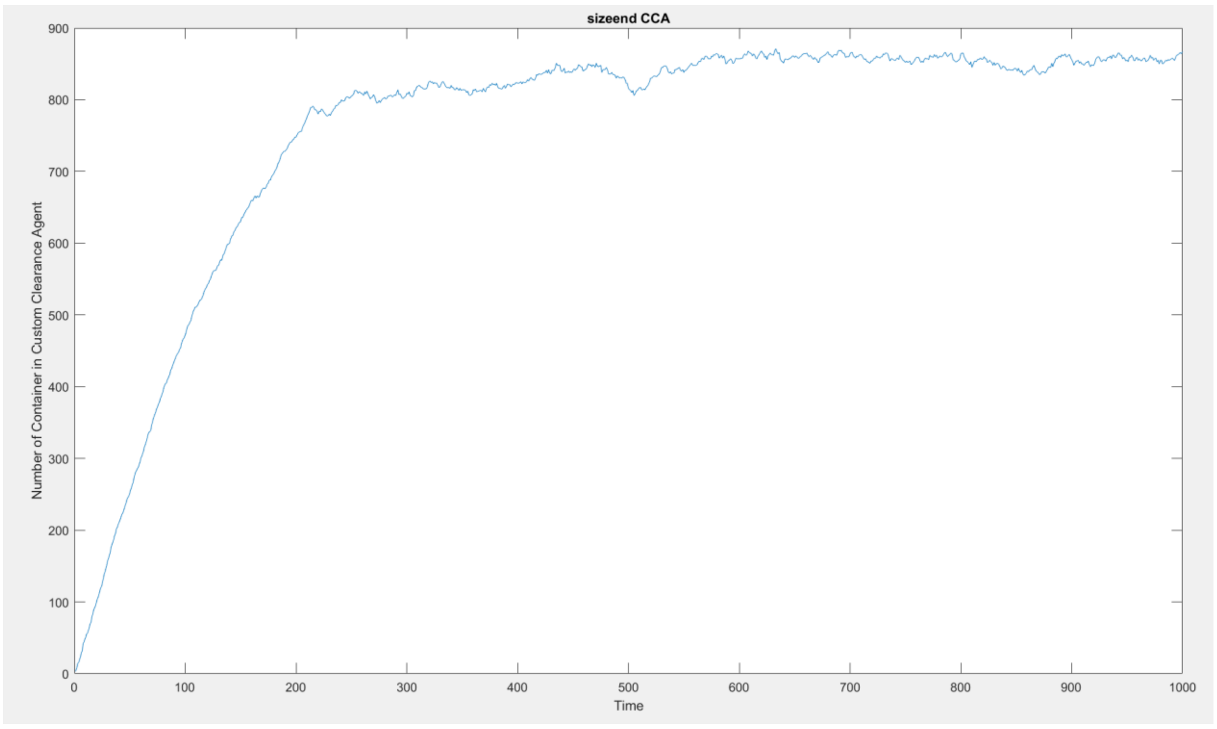


Figure 117 Sizeend of the CCA Agent - 1000 its ens_max 10 and work max 10

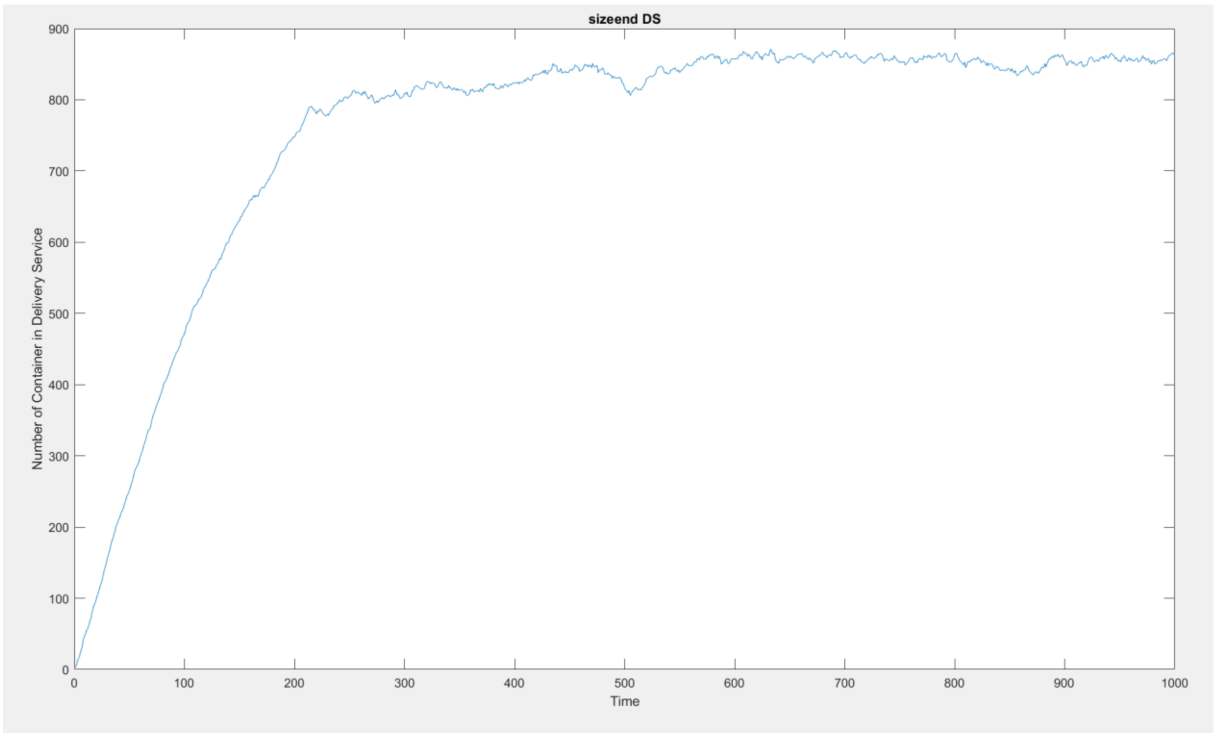


Figure 118 Sizeend of the Delivery Service Agent - 1000 its ens_max 10 and work max 10

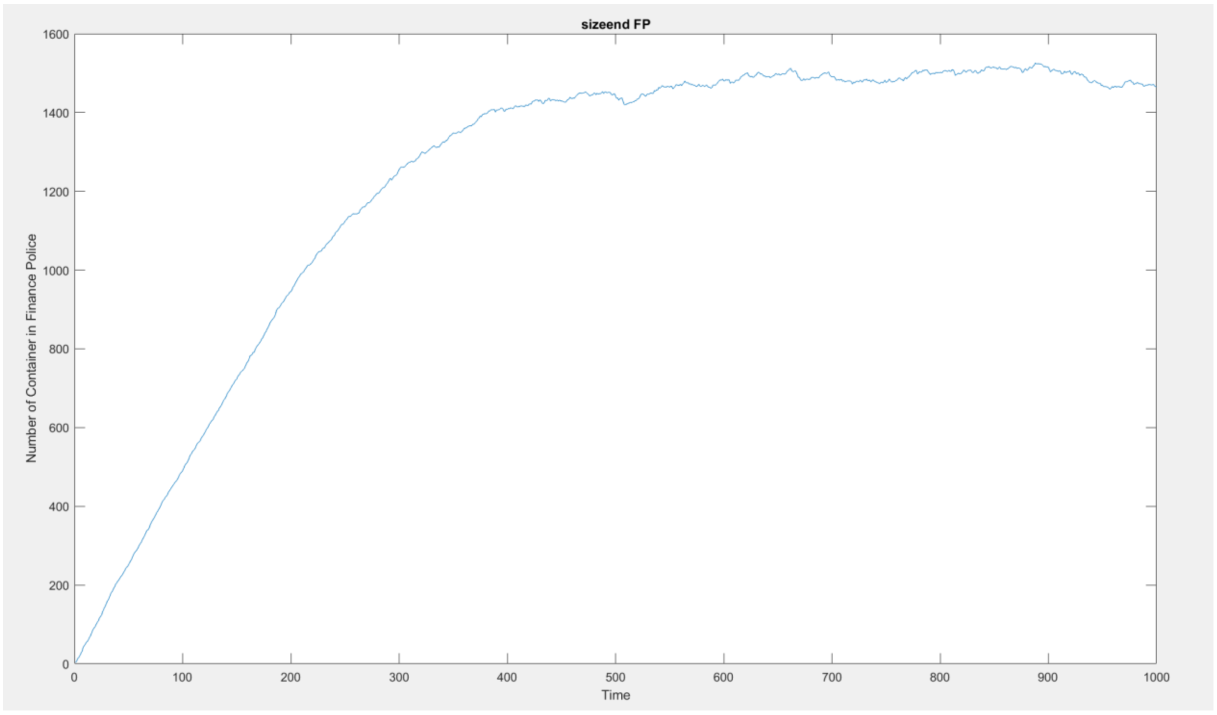


Figure 119 Sizeend of the Finance Police Agent - 1000 its ens_max 10 and work max 10

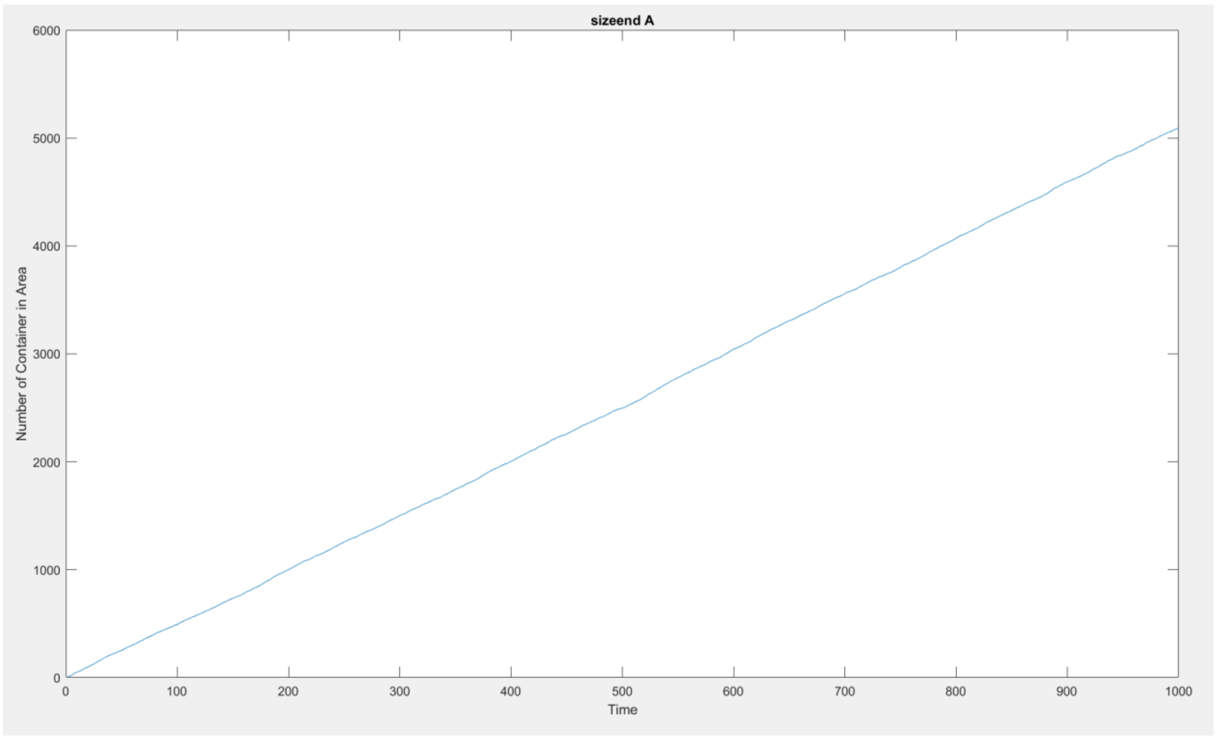


Figure 120 Sizeend of the Area Agent - 1000 its ens_max 10 and work max 10

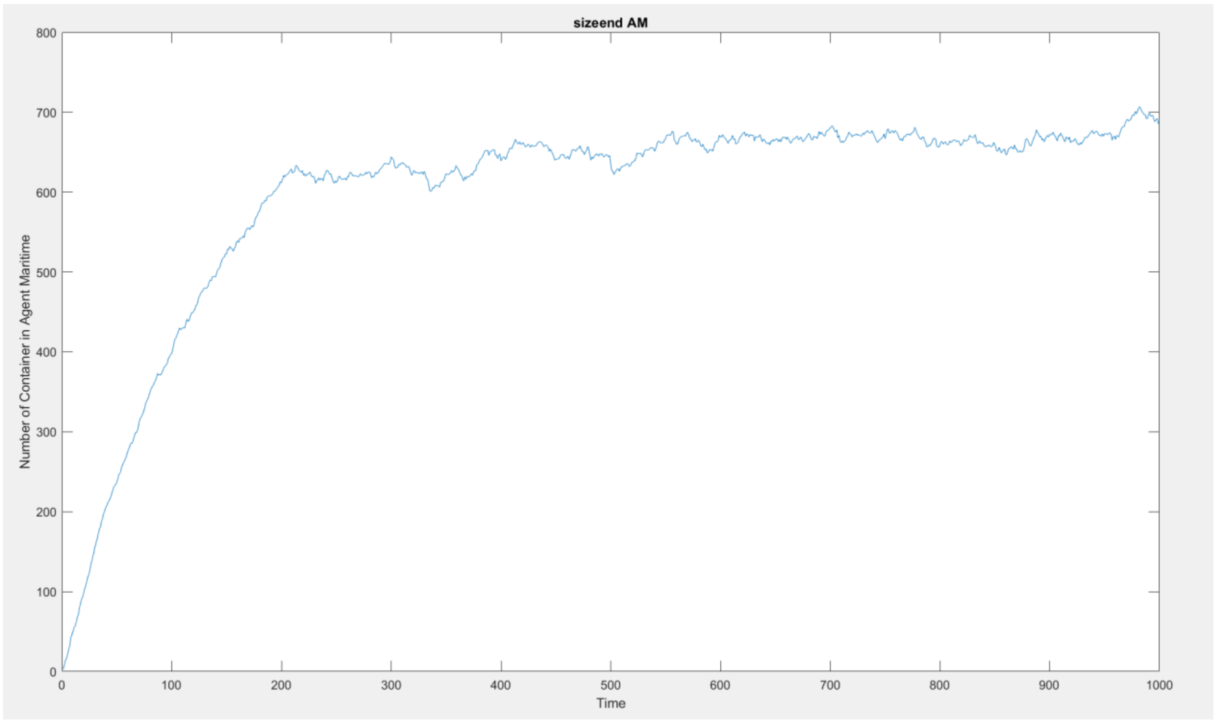


Figure 121 Sizeend of the Agent Maritime- 1000 its ens_max 10 and work max 10

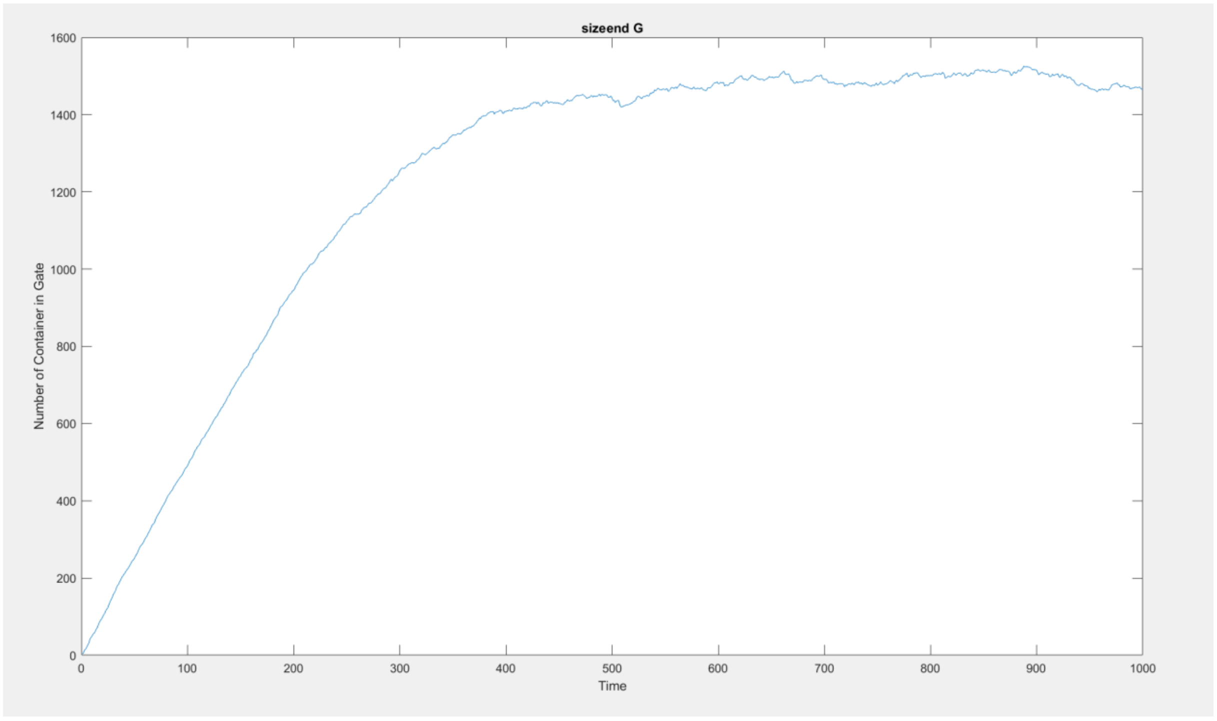


Figure 122 Sizeend of the Gate Agent - 1000 its ens_max 10 and work max 10

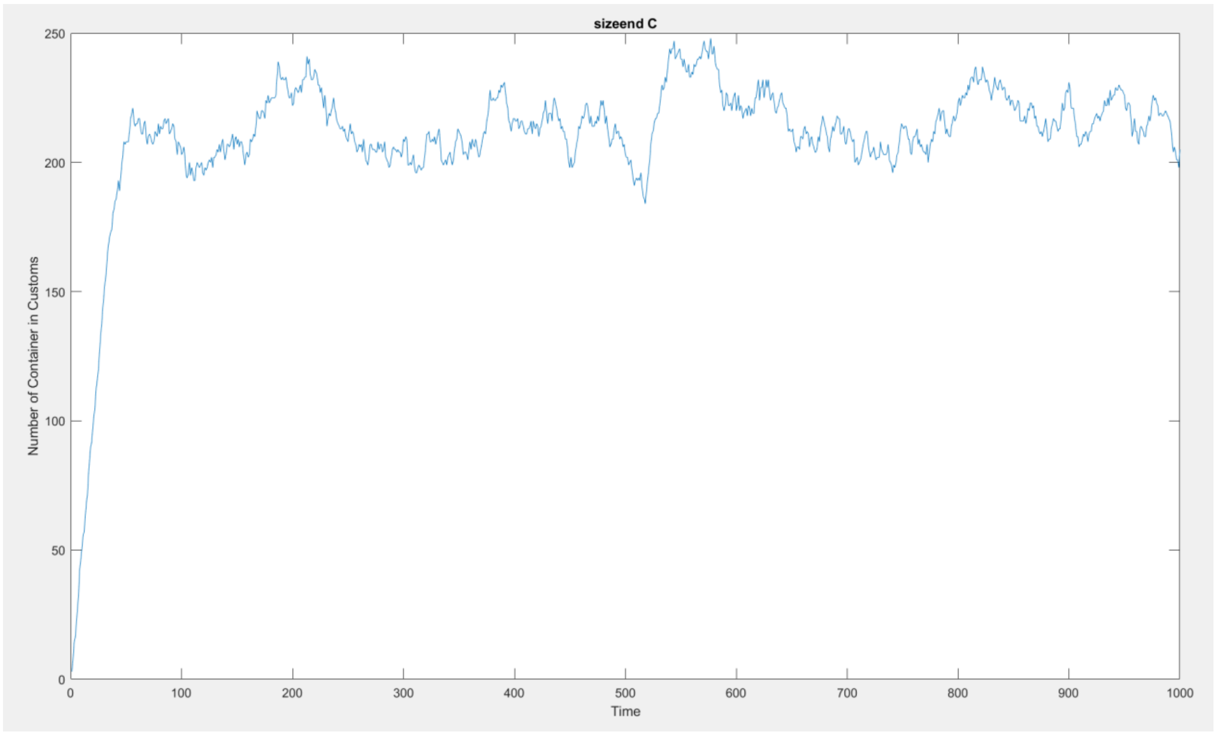


Figure 123 Sizeend of the Customs Agent - 1000 its ens_max 10 and work max 10

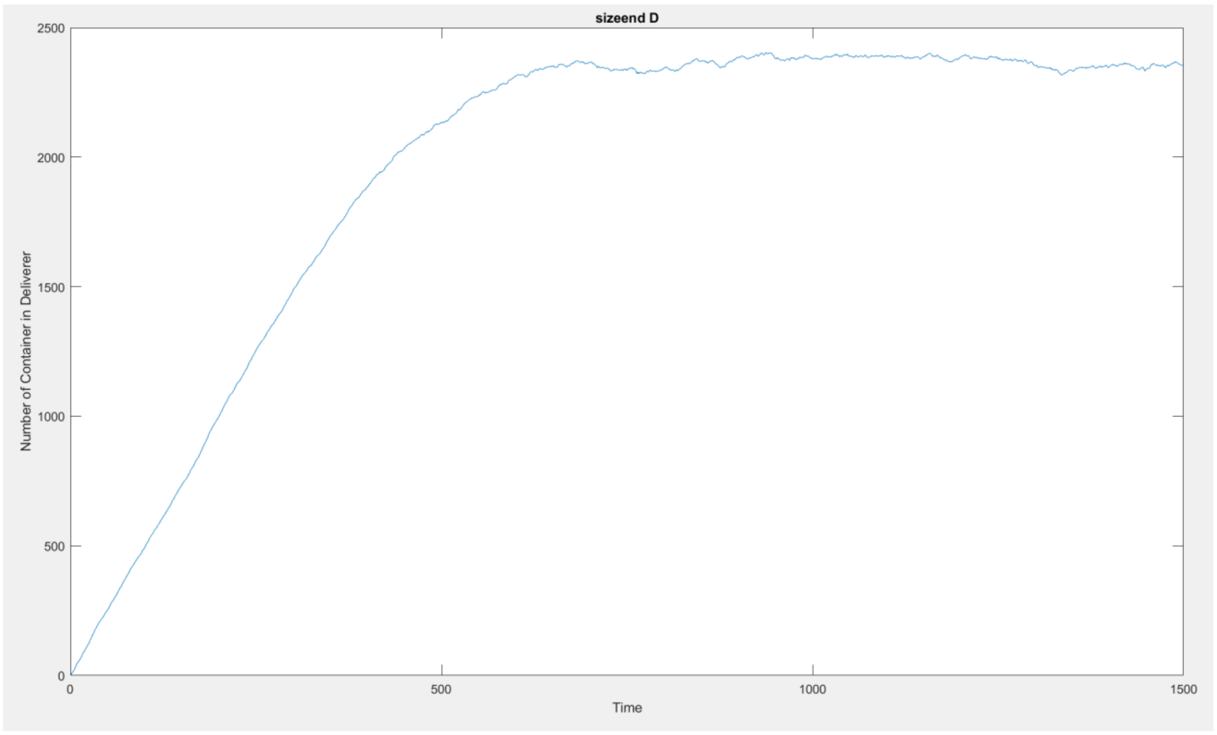


Figure 124 Sizeend of the Deliverer Agent - 1500 its ens_max 10 and work max 10

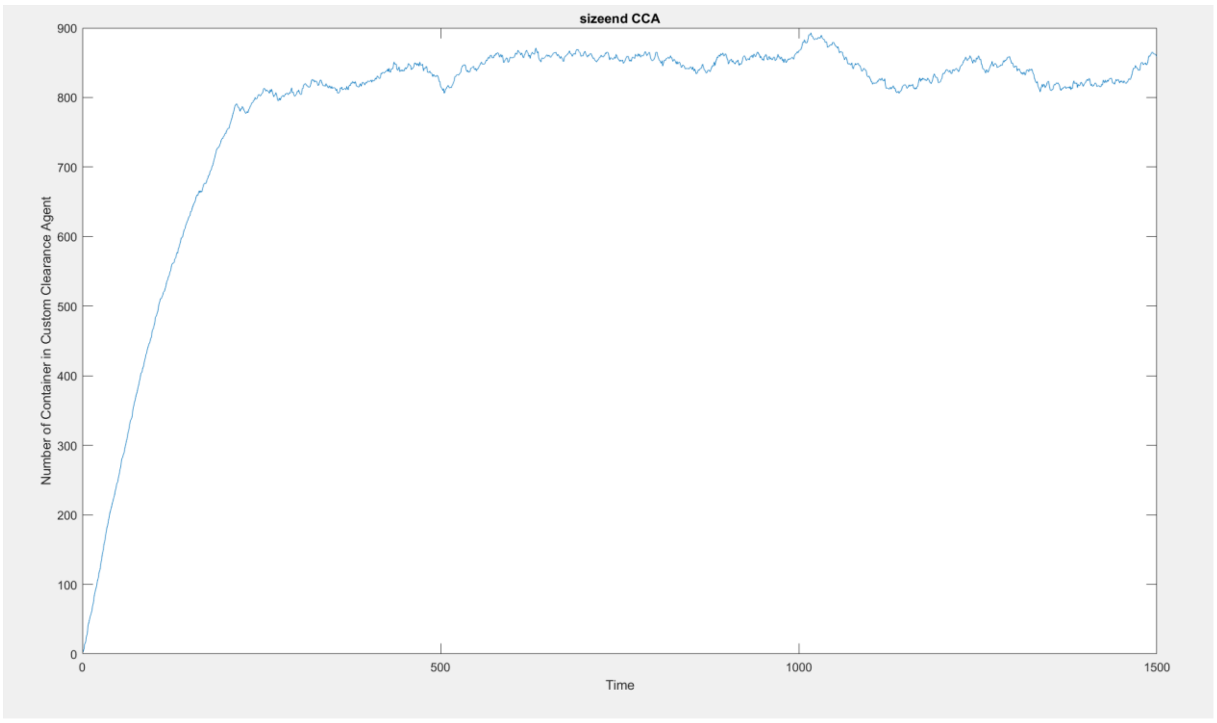


Figure 125 Sizeend of the CCA Agent - 1500 its ens_max 10 and work max 10

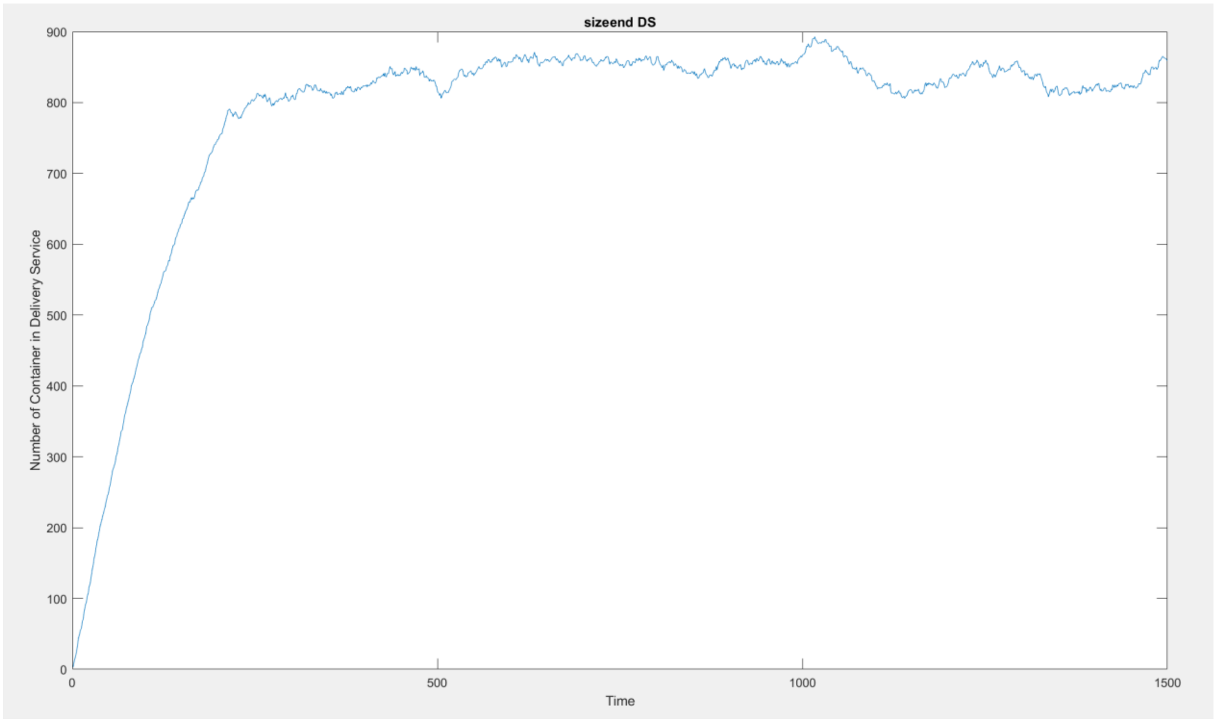


Figure 126 Sizeend of the Delivery Service Agent - 1500 its ens_max 10 and work max 10

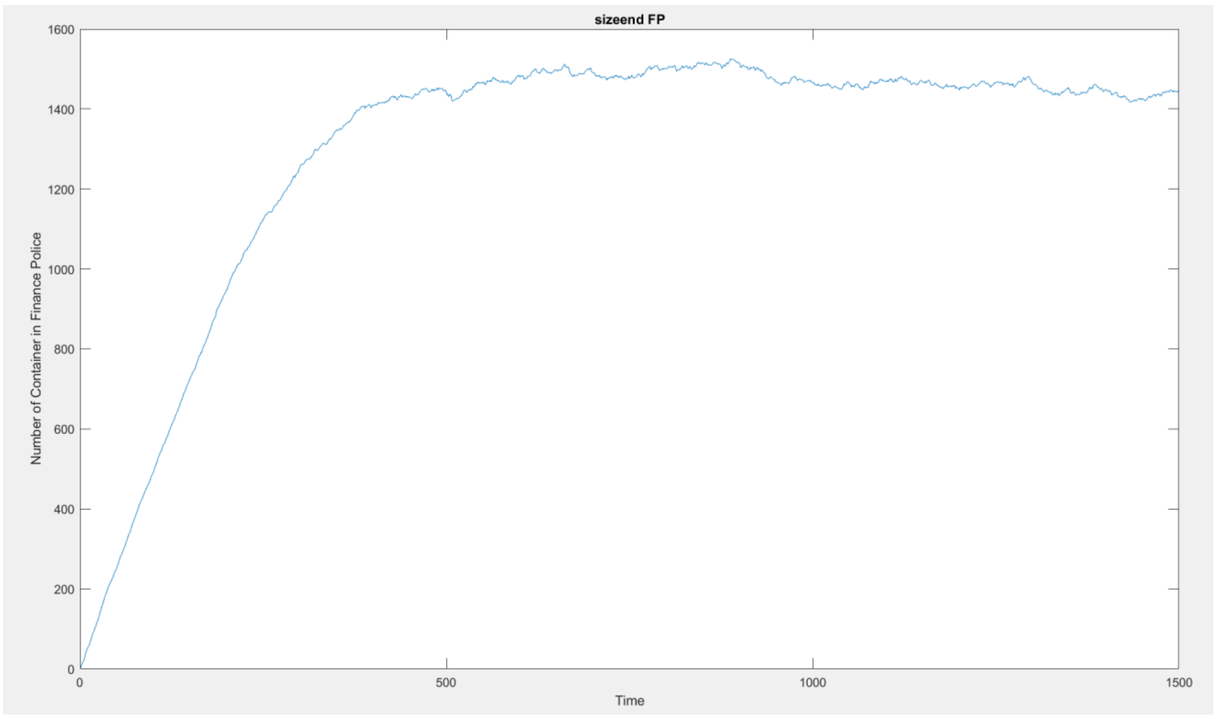


Figure 127 Sizeend of the Finance Police Agent - 1500 its ens_max 10 and work max 10

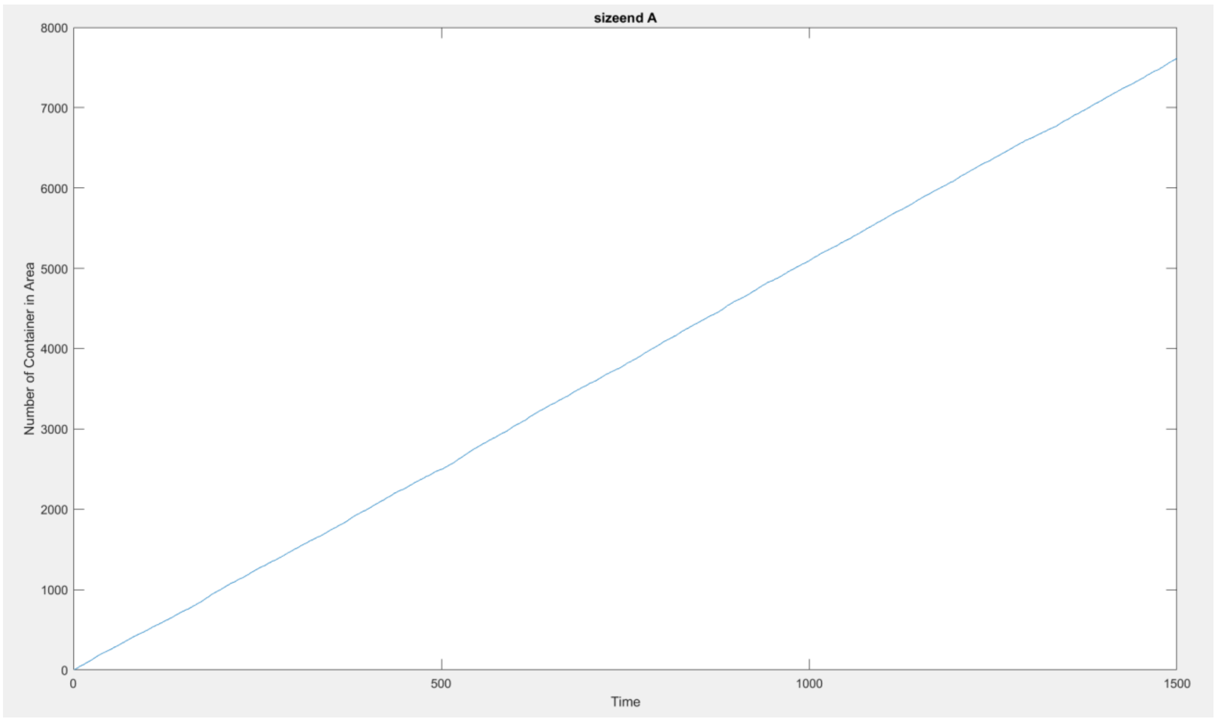


Figure 128 Sizeend of the Area Agent - 1500 its ens_max 10 and work max 10

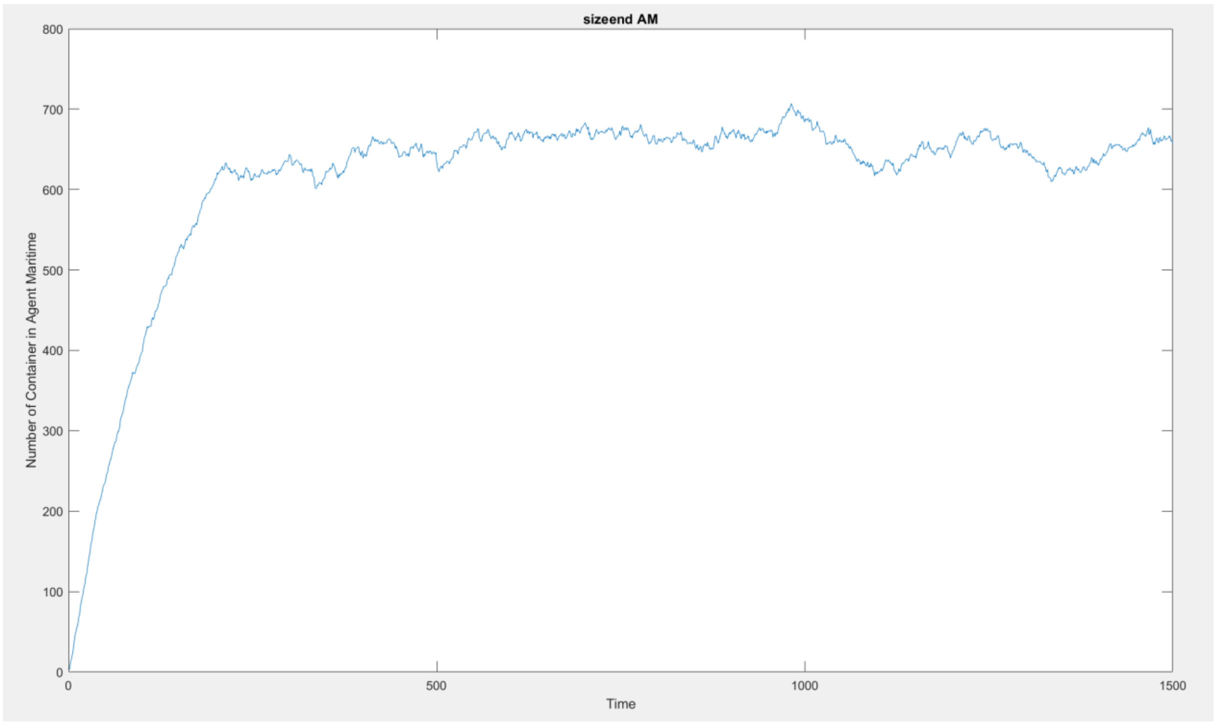


Figure 129 Sizeend of the Agent Maritime- 1500 its ens_max 10 and work max 10

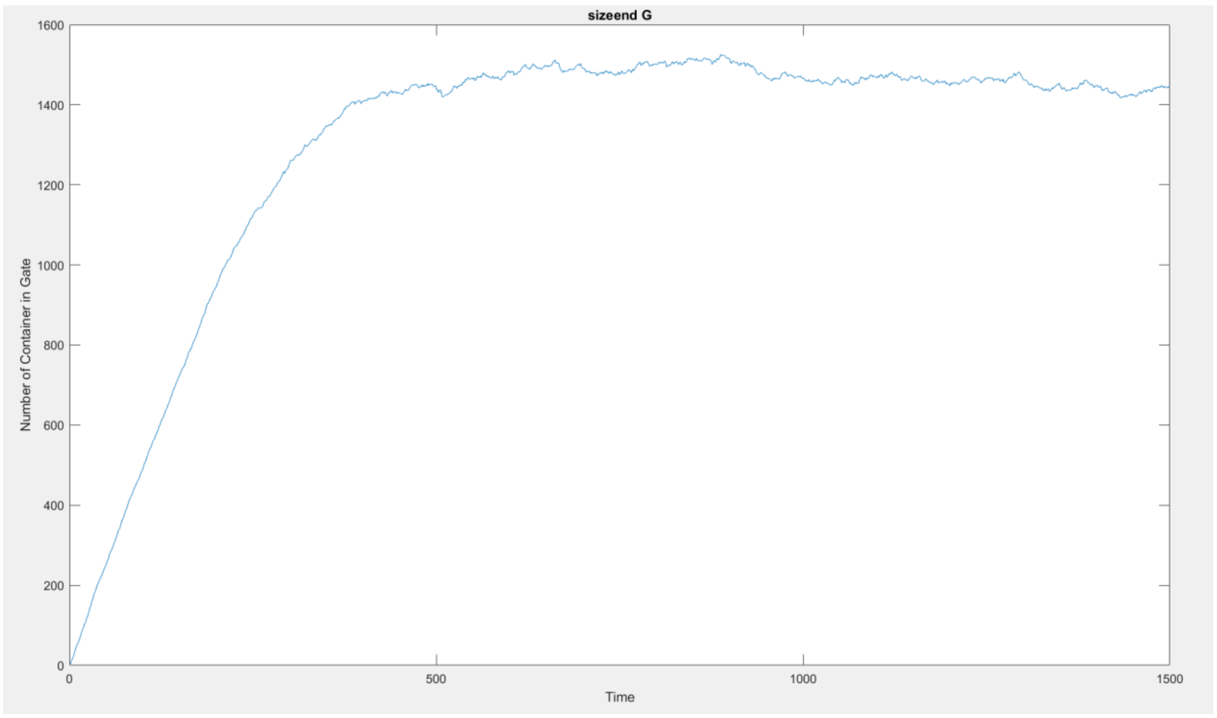


Figure 130 Sizeend of the Gate Agent - 1500 its ens_max 10 and work max 10

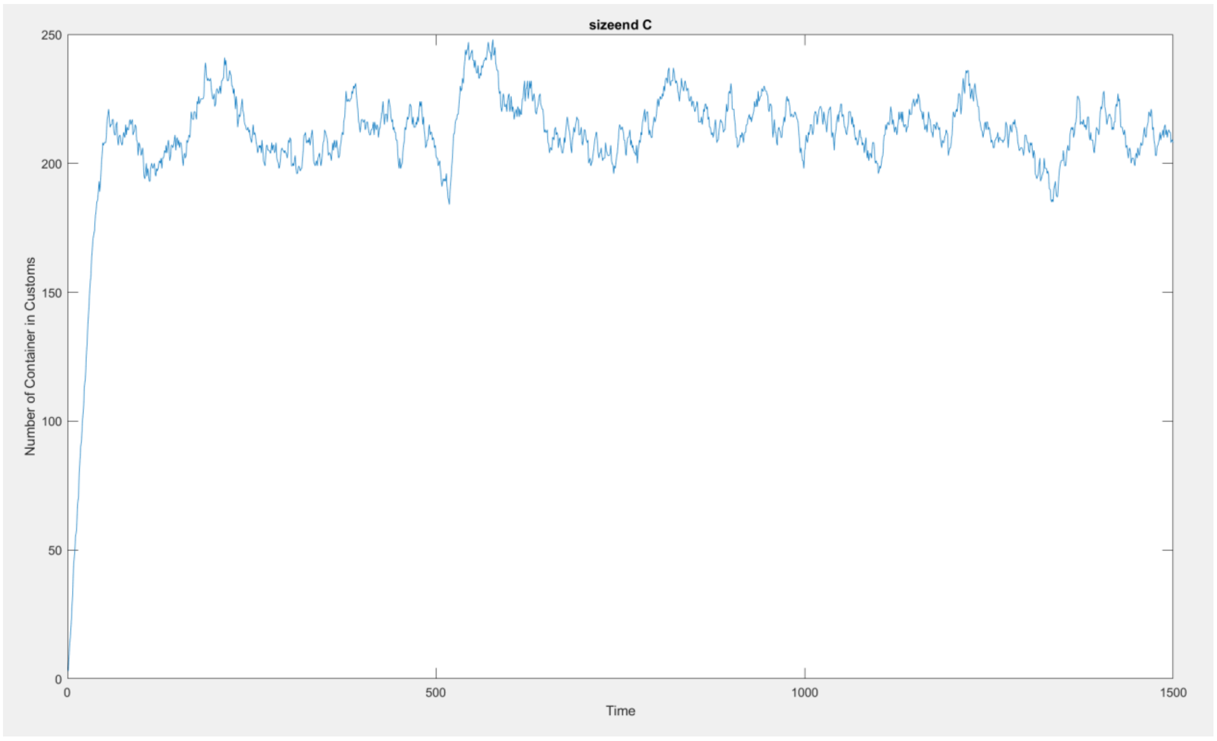


Figure 131 Sizeend of the Customs Agent - 1500 its ens_max 10 and work max 10

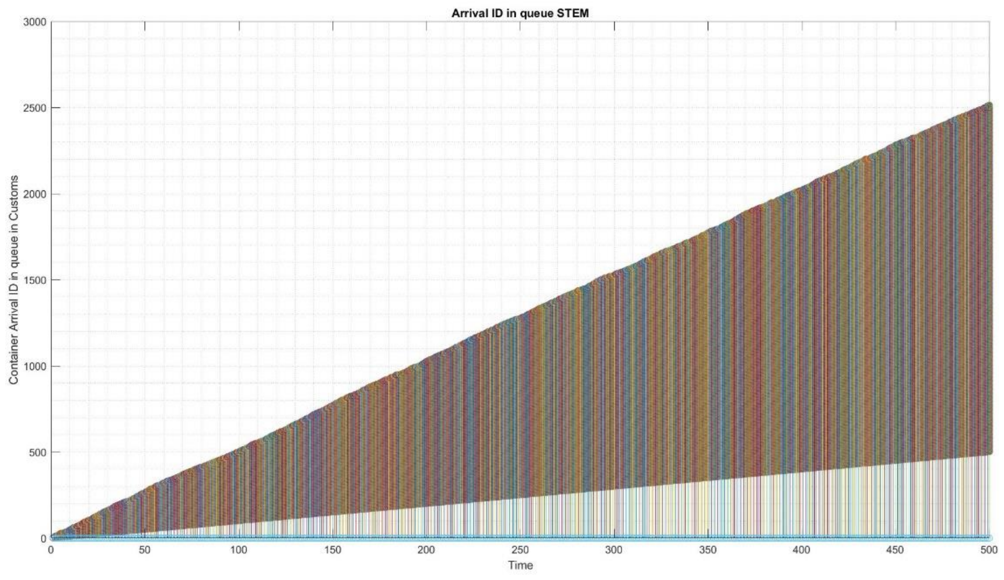


Figure 132 Sequence of container in the Customs queue – 500 iterations ens_max 10 and work max 1

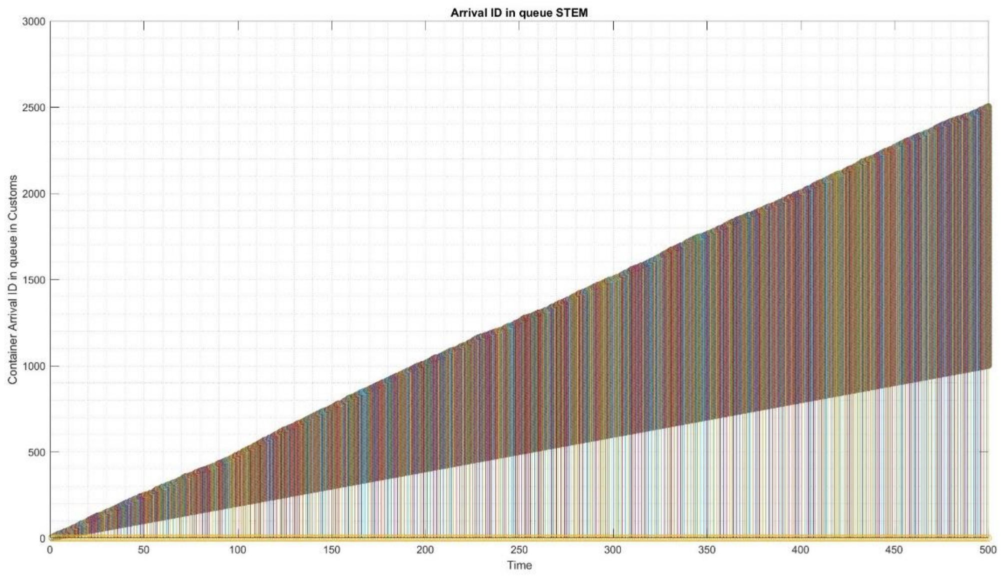


Figure 133 Sequence of container in the Customs queue – 500 iterations ens_max 10 and work max 2

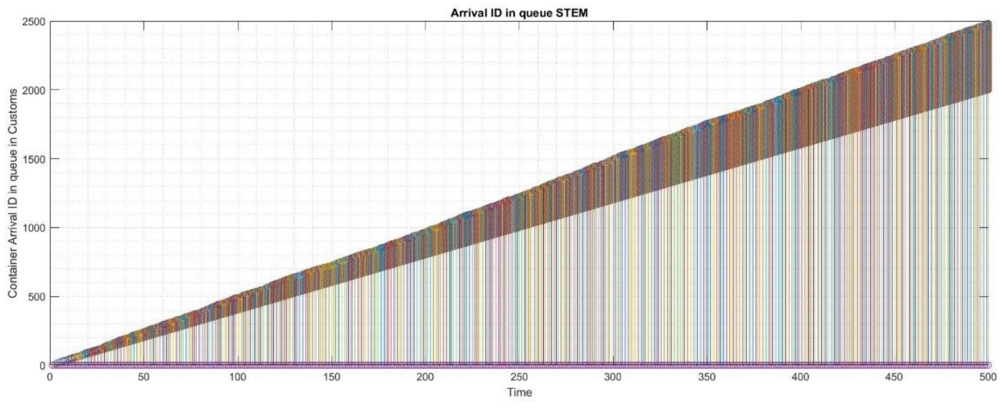


Figure 134 Sequence of container in the Customs queue – 500 iterations ens_max 10 and work max 4

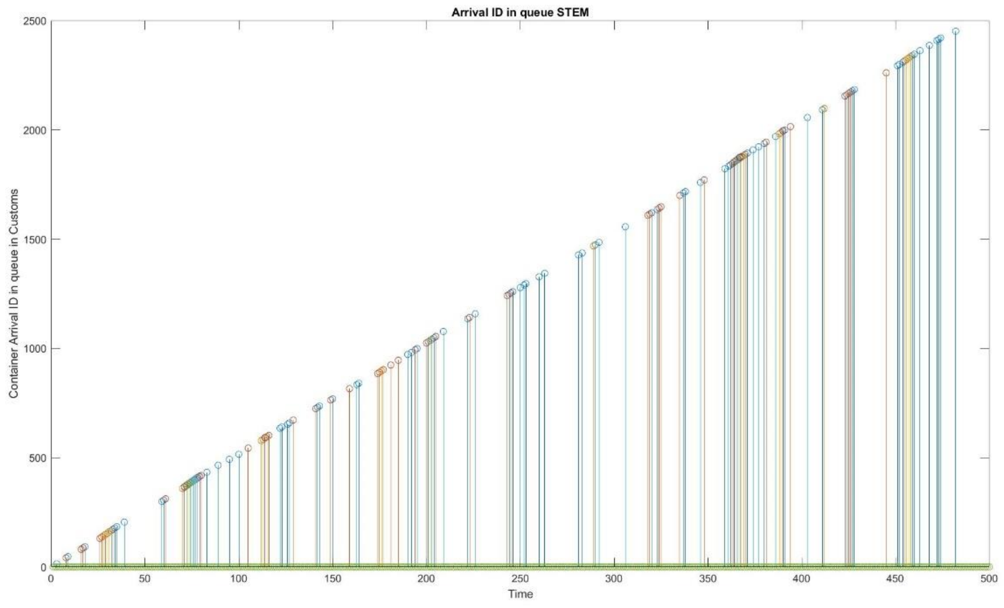


Figure 135 Sequence of container in the Customs queue – 500 iterations ens_max 10 and work max 6

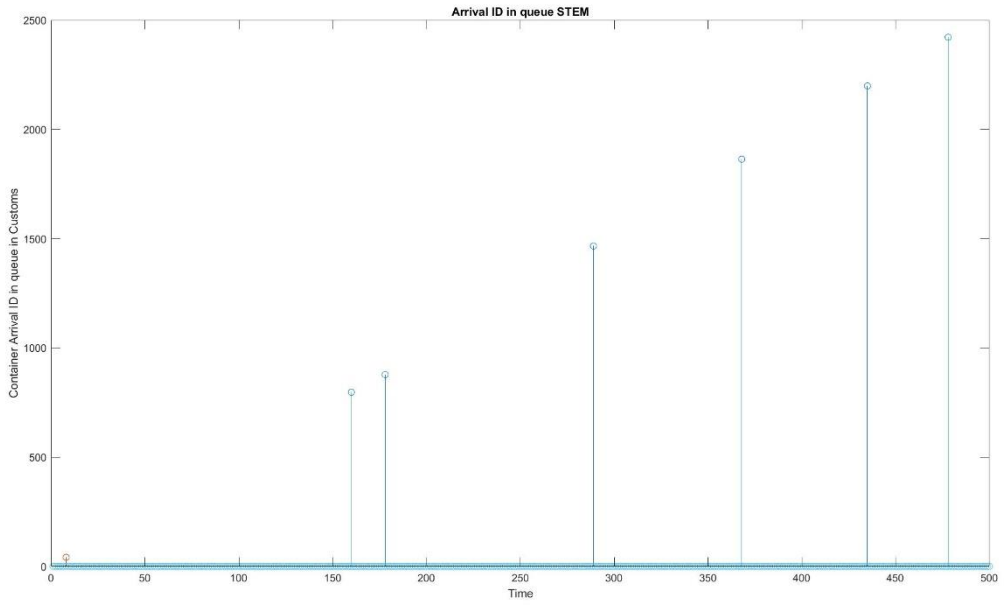


Figure 136 Sequence of container in the Customs queue – 500 iterations ens_max 10 and work max 8

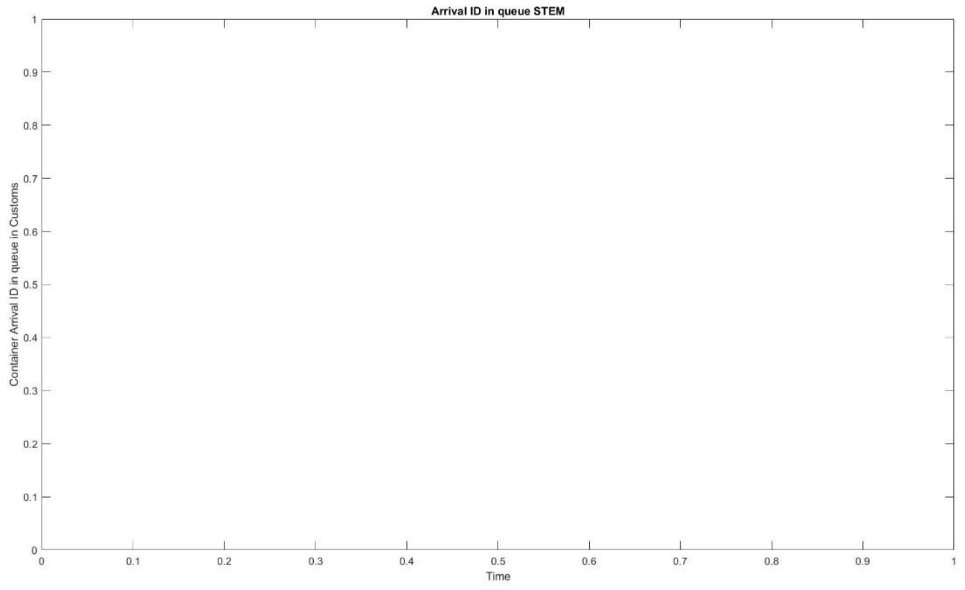


Figure 137 Sequence of container in the Customs queue – 500 iterations ens_max 10 and work max 10

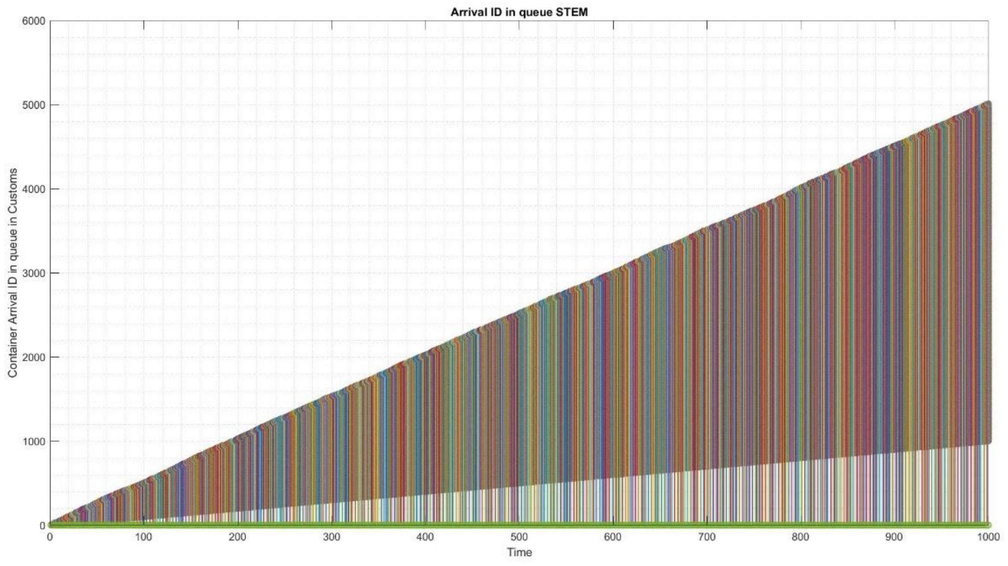


Figure 138 Sequence of container in the Customs queue – 1000 iterations ens_max 10 and work max 1

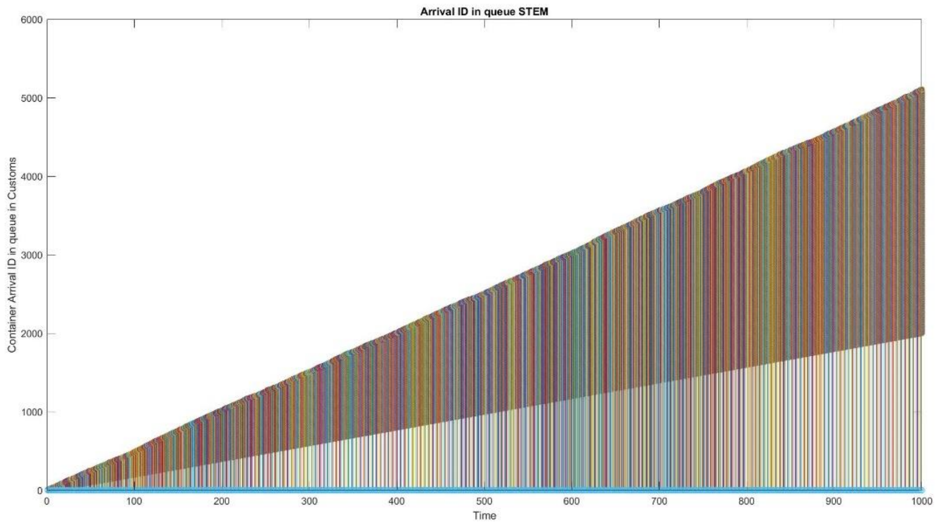


Figure 139 Sequence of container in the Customs queue – 1000 iterations ens_max 10 and work max 2

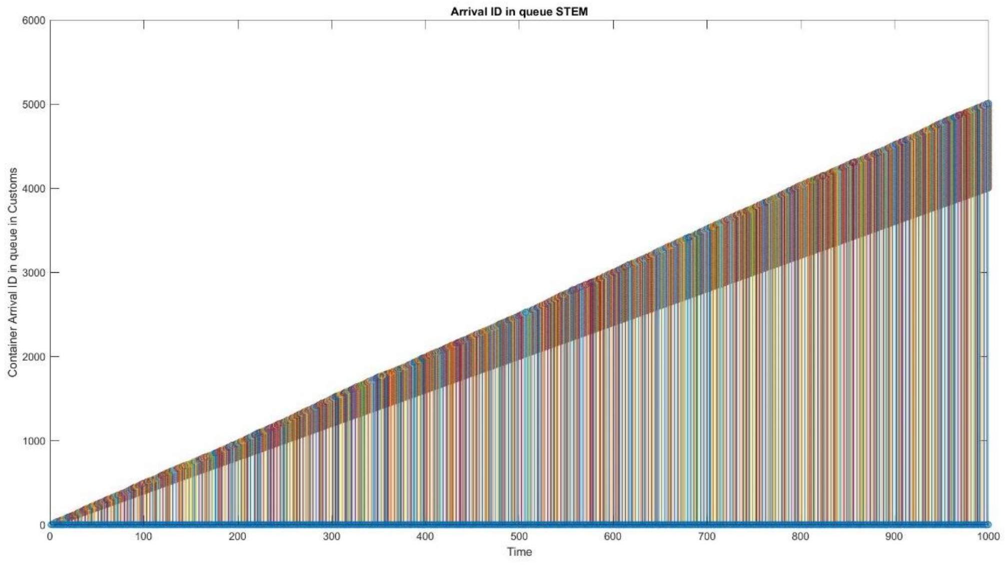


Figure 140 Sequence of container in the Customs queue – 1000 iterations ens_max 10 and work max 4

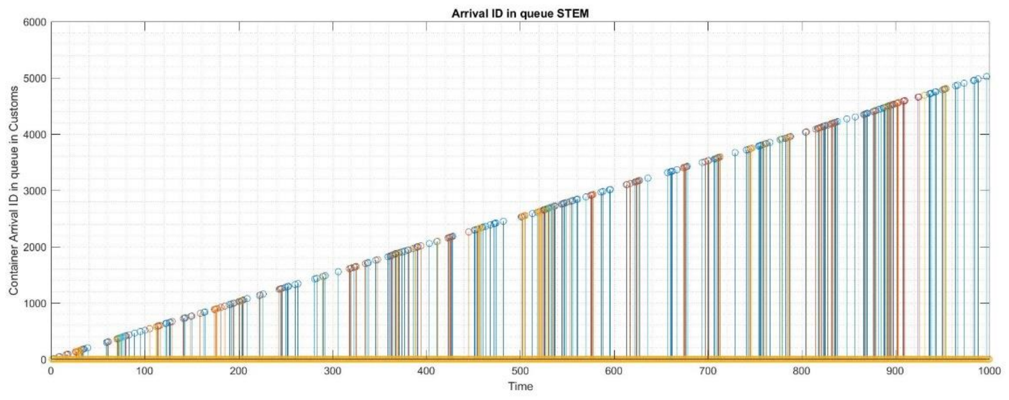


Figure 141 Sequence of container in the Customs queue – 1000 iterations ens_max 10 and work max 6

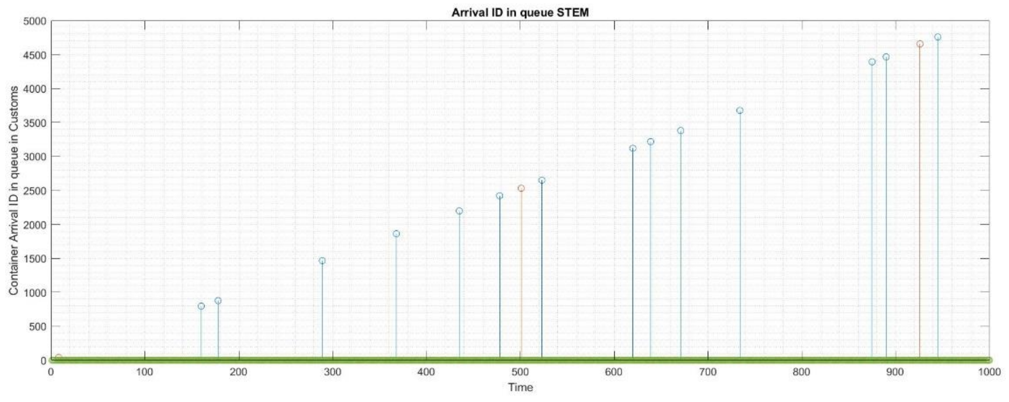


Figure 142 Sequence of container in the Customs queue – 1000 iterations ens_max 10 and work max 8

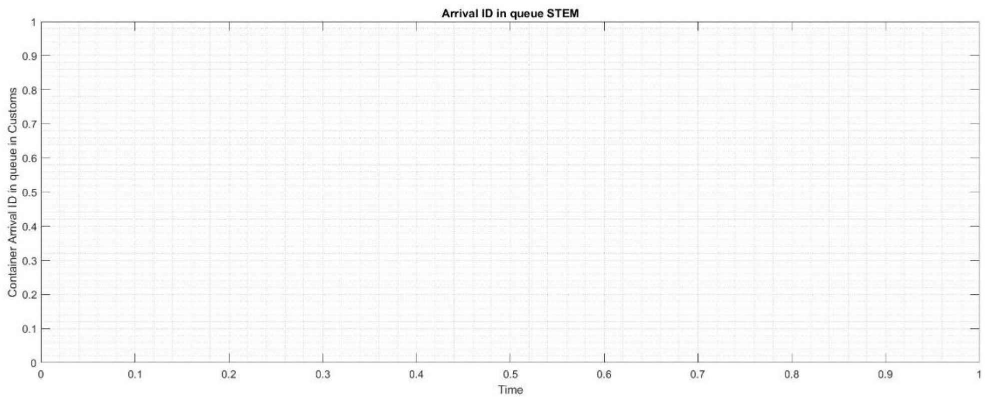


Figure 143 Sequence of container in the Customs queue – 1000 iterations ens_max 10 and work max 10

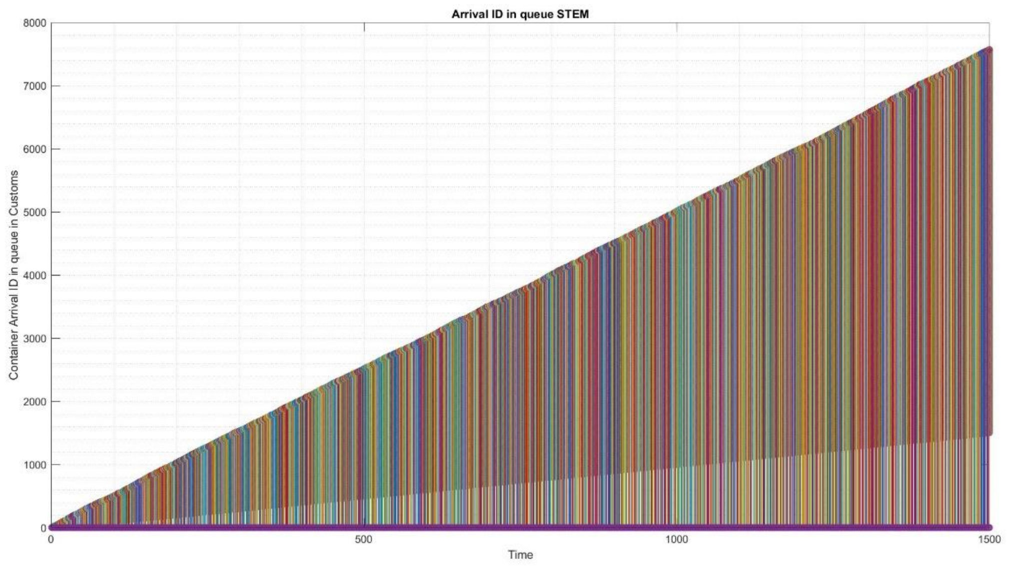


Figure 144 Sequence of container in the Customs queue – 1500 iterations ens_max 10 and work max 1

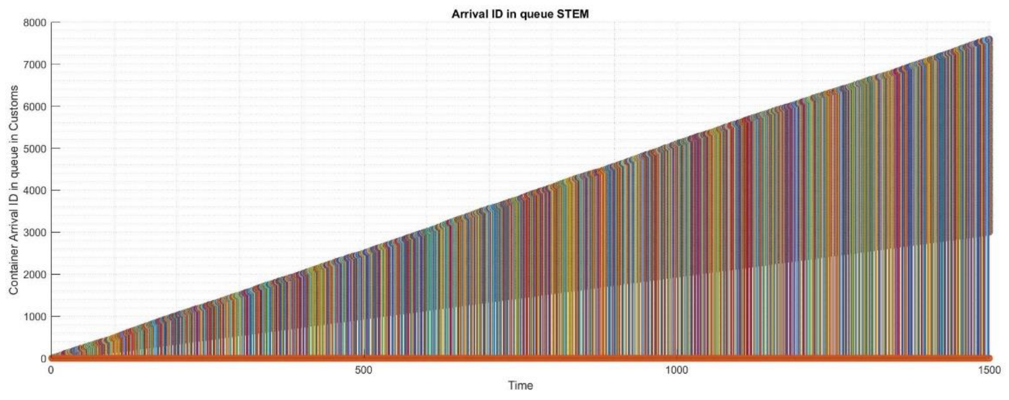


Figure 145 Sequence of container in the Customs queue – 1500 iterations ens_max 10 and work max 2

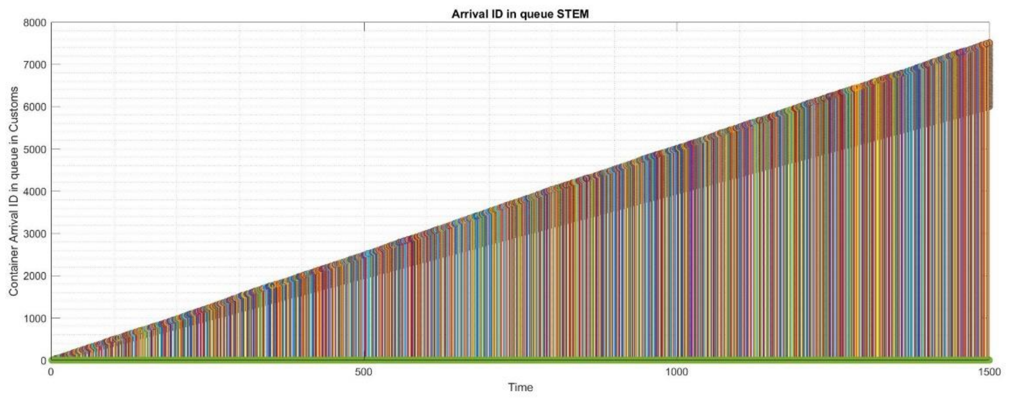


Figure 146 Sequence of container in the Customs queue – 1500 iterations ens_max 10 and work max 4

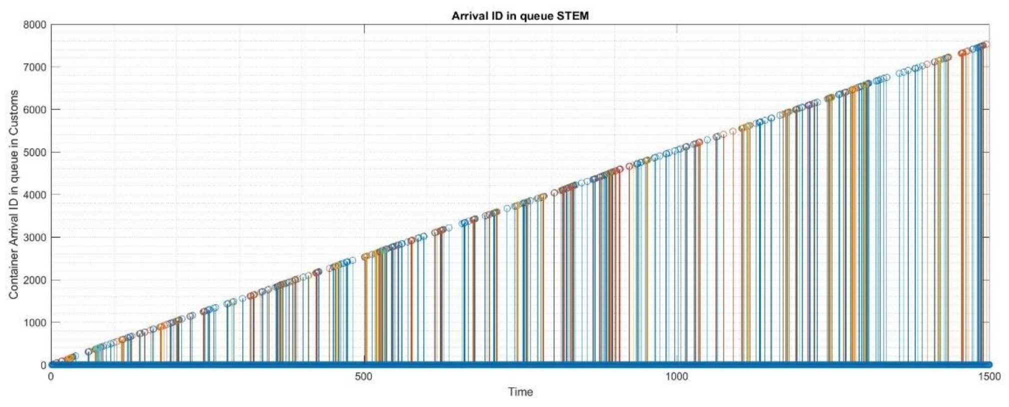


Figure 147 Sequence of container in the Customs queue – 1500 iterations ens_max 10 and work max 6

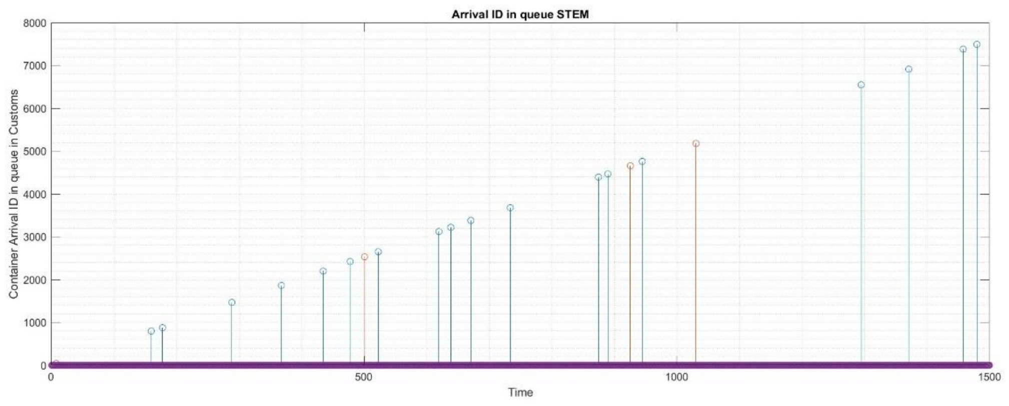


Figure 148 Sequence of container in the Customs queue – 1500 iterations ens_max 10 and work max 8

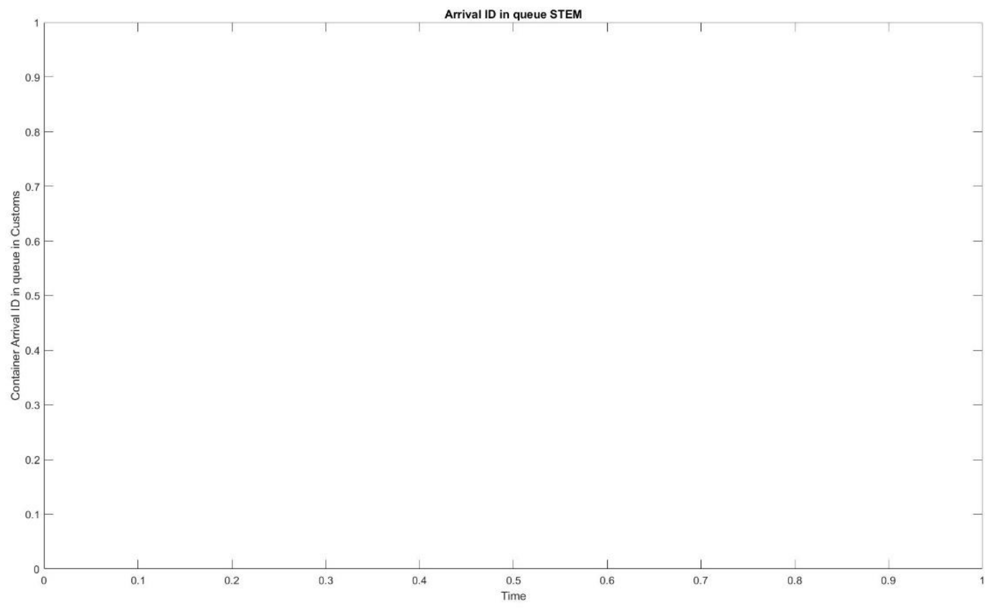


Figure 149 Sequence of container in the Customs queue – 1500 iterations ens_max 10 and work max 10

Bibliography

[1] Abar S, et al. (2007) Agent Based Modelling and Simulation tools: A review of the state-of-art software", Computer Science Review. Computer Science ReviewVolume 24, Pages 13-33.

[2] Arthur W.B, Holland J. H, LeBaron B. D, Palmer R, Tayler P. (1997) Asset pricing under endogeneous expectations in an artificial stock market. The Economy as an Evolving Complex System II, ser. SFI Studies in the Sciences of Complexity, W. Arthur, S. Durlauf, and D. Lane, Eds. Addison Wesley Longman.

[3] Astesiano E, Reggio G, Ricca F. (2008). Modeling business within a uml-based rigorous software development approach. Lecture Notes in Computer Science, vol. 5065, pp. 261–277.

[4] Balsamo S., UNIVE. URL http://www.dsi.unive.it/~balsamo/disp.pdf/Cap1.pdf

[5] Becker J, Kugeler M, Rosemann M. (2013) Process management: a guide for the design of business processes. Springer Science & Business Media.

[6] Bianchi C, Cirillo P, Gallegati M, and Vagliasindi P.A. (2007) Validating and calibrating agent-based models: a case study. Computational Economics, vol. 30, no. 3, pp. 245–264.

[7] Bohl G, Harting P, Sander van der Hoog. (2017) ETACE Virtual Appliance User Manual. Bielefeld University, Chair for Economic Theory and Computational Economics (ETACE).

[8] Bohl G, Harting P, Sander van der Hoog. (2014) The ETACE Virtual Appliance: An Exploratory for the Eurace@Unibi model.

[9] Caliri G.V. Introduction to Analytical Modeling. BMC Software, Inc. Waltham MA USA.

[10] Chih-Hao L, Chung-Yao K, Chong-Ye H. (2015) Managing emergency department overcrowding via ambulance diversion: A discrete event simulation model, Journal of the Formosan Medical Association 114, 64-71.

[11] Cincotti, S., Raberto, M., & Teglio, A. (2010). Credit money and macroeconomic instability in the agent-based model and simulator Eurace. Economics: The Open- Access, Open-Assessment E-Journal, 4 (2010-26).

[12] Cincotti, S., Raberto, M., & Teglio, A. (2012a). The Eurace macroeconomic model and simulator. In Agent-based Dynamics, Norms, and Corporate Governance. The proceedings of the 16-th World Congress of the International Economic Association, volume II. Palgrave.

[13] Cincotti, S., Raberto, M., & Teglio, A. (2012b). Macroprudential policies in an agent-based artificial economy. Agent-based Models and Economy Policy, 205–234.

[14] Cincotti S, Petrović M, Ozel B, Andrea Teglio A, Raberto M. (2017) Eurace Open: An agent-based multi-country model.

[15] Cincotti S, Raberto M, Focardi S.M, Marchesi M. (2001) Agent-based simulation of a financial market. Physica A 299 No. 1-2. pp. 320-328.

[16] Coakley S, Gheorghe M, Holcombe M, Chin S, D. Worth, and C. Greenough. (2012). Exploitation of High Performance Computing in the FLAME Agent-Based Simulation Framework, in Proceedings of the 14th International Conference on High Performance Computing and Communications (HPCC2012), Liverpool, UK, New York: IEEE, 2012 pp. 538–545. doi:10.1109/HPCC.2012.79.

[17] Coakley S, Smallwood R and Holcombe M. (2006) Using X-Machines as a formal basis for describing agents in agent-based modelling, Proceedings of the 2006 Agent-Directed Simulation Conference, 33-40.

[18] Coakley S, Smallwood R and Holcombe M., Rod Smallwood. A General Framework for agent-based modelling of complex systems, Proceedings of the 2006 European Conference on Complex Systems.

[19] Crooks Andrew T, Christian J. E. Castle. (2006). Principles and Concepts of Agent-Based Modelling for Developing Geospatial Simulations.

[20] Davenport T.H. (1993). Process innovation: reengineering work through information technology. Harvard Business Press.

[21] Davidsson P, Henesey L, Ramstedt L, Törnquist J., and Wernstedt F., An Analysis of Agent-Based Approaches to Transport Logistics. Transportation Research Part C: Emerging Technologies, Vol. 13(4), pp. 255-271, Elsevier, 2005.

[22] Davidsson P, Henesey L, Ramstedt L, Törnquist J, and Wernstedt F, (2005) Agent-Based Approaches to Transport Logistics. Applications of Agent Technology in Traffic and Transportation: Whitestein Series in Software Agent Technologies.

[23] De Sousa R. T, De Deus F, Aires B, Ribeiro G, Araujo A. P, Holanda M, Vidal S. S, Dos Santos R. M, Cortes F. P. (2014). Business process modelling: A study case within the Brazilian ministry of planning, budgeting and management," in Information Systems and Technologies (CISTI), 2014 9th Iberian Conference on. IEEE, pp. 1–6.

[24] Delgado A, Ruiz F, I. G.-R. de Guzman, and Piattini M. (2010). Minerva: model driven and service oriented framework for the continuous business process improvement and related tools in Service-Oriented Computing. ICSOC/ServiceWave Workshops. Springer, pp. 456–466.

[25] Donatelli M. (1995). Sistemi nella gestione integrata delle colture - Appunti delle lezioni. Pubblicazione speciale dell'Istituto Sperimentale Agronomico,ISA-Sezione di Modena,Modena. URL: http://www.isci.it/isciTC/references/sistemi/cap_2.pdf

[26] Dosi et al., 2010, Assenza et al., 2015, Hommes, 2013, and the survey Fagiolo and Roventini, 2017

[27] Downing T.E, Moss S, Pahl-Wostl C. (2000) Understanding climate policy using participatory agent-based social simulation. International Workshop on Multi-Agent Systems and Agent-Based Simulation. Springer, pp. 198–213.

[28] Dresner K, Stone P, (2008). A multiagent approach to autonomous intersection management," Journal of artificial intelligence research, vol. 31, pp. 591–656.

[29] EURACE Stategraph. (2013) http://www.dcs.shef.ac.uk/~wmlh/stategraph.pdf.

[30] EURACE at the University of Sheffield. (2013) http://www.eurace.group.shef.ac.uk.

[31] "Fakultät für Wirtschaftswissenschaften." (2013) http://www.wiwi.uni-bielefeld.de/vpl1/projects/eurace.html.

[32] ETACE Virtual Appliance Download, Accessed: 21.03.2014. http://pub. uni-bielefeld.de/data/2674041.

[33] ETACE Virtual Appliance Website, Accessed: 21.03.2014. http://www.wiwi.uni-bielefeld.de/ lehrbereiche/vwl/etace/Eurace Unibi/Virtual Appliance.

[34] Fagiolo G, Guerini M, Lamperti F, Moneta A, Roventini A, and Sapio A. (2017). Validation of agent-based models in economics and finance. Laboratory of Economics and Management (LEM), Sant'Anna School of Advanced Studies, Pisa, Italy, Tech. Rep.

[35] FLAME: Flexible Large-Scale Agent Modelling Environment. (2013) http://www.flame.ac.uk.

[36] Genoa Port Center, Terminal of the Container. URL: http://www.genoaportcenter.it/ Pagina.asp x?idPag=110

[37] Genovese A, Barbati, Bruno G. (2011). Applications of agent-based models for optimization problems: a literature review. Lecture Notes in Management Science.

[38] GESTEC - Service Oriented Technologies for Integrated ICT platforms (Italian Ministry for Education Research and University, DM 64565.

[39] Giglio D, UNIGE Modelli e Metodi per l'Automazione" del Corso di Laurea in Ingegneria Gestionale A.A. 12/13. URL: http://www.dist.unige.it/giglio/didattica/mmafiles/files/08S_ Simulazione.pdf

[40] Gilbert N, Troitzsch K.G. (2005). Simulation for the Social Scientist, second ed. Open University Press, Berkshire, UK.

[41] Harrington H. J, Harrington J. S. (1995). Total improvement management: the next generation in performance improvement. McGraw-Hill Professional.

[42] Held P. F, Ian F. Wilkinson, Robert E. Marks, Young L. (2014). Agent-based Modelling, a new kind of research.

[43] Henesey, L. A Review of Decision Support Systems in Container Terminal Operations.

[44] Henesey, L., Notteboom, T., and Davidsson, P. (2003). Agent-based simulation of stakeholders relations: An approach to sustainable port and terminal management. Proceedings of the International Association of Maritime Economists Annual Conference, Busan, Korea.

[45] Henesey, L., Davidsson, P., and Persson, J.A. (2006). Evaluating Container Terminal Transhipment Operational Policies: An Agent-Based Simulation Approach. Journal WSEAS Transactions on Computers Vol. 5(9), pp. 2090-2098.

[46] Henesey, L. and Persson, J.A. (2006). Application of Transaction Costs In Analyzing Transport Corridors Using Multi-Agent-Based Simulation. Extended version of article published in Promet Traffic & Transportation: Scientific Journal on Traffic and Transportation Research. Vol. 18(2), pp. 59-65.

[47] Henesey, L. (2003). More than just Piers: a multi-agent system in defining organization in a seaport terminal management system" Proceedings of the 47th Annual Conference of the International Society for the Systems Sciences (ISSS) (Special Integration Group on Systems Applications to Business and Industry), Crete, Greece.

[48] Henesey, L. (2004). A Multi Agent Based Simulator for Managing a Container Terminal. Proceedings of the 2nd European Workshop on Multi-Agent Systems (EUMAS 2004), Barcelona, Spain, pp. 291-302.

[49] Huang Y.-S, Weng Y.-S, Zhou M. (2014) Modular design of urban traffic-light control systems based on synchronized timed petri nets. IEEE Transactions on Intelligent Transportation Systems, vol. 15, no. 2, pp. 530–539.

[50] Holcombe M, Coakley S, Kiran M, Chin S, Greenough C, Worth D, Cincotti S, Raberto M, Teglio A, Deissenberg, Sander van der Hoog, Herbert Dawid, Simon Gemkow, Philipp Harting, Michael Neugart. Large-Scale Modeling of Economic Systems.

[51] URL:http://www.bpmn.org

[52] URL: http://www.economics-ejournal.org/economics/discussionpapers/2010-4.

[53] URL:http://www.genoaportcenter.it/Pagina.aspx? idMss=74&idPag=191& AspxAuto Detect CookieSupport=1

[54] URL: http://iceace.github.io/home/

[55] URL: https://whatis.techtarget.com/definition/pseudocode.

[56] URL: http://www.agentbuilder.com/ Documentation/whyAgents.html

[57] Iovanella A. Università di Roma Tor Vergata, Dipartimento di Ingegneria dell'Impresa, Facoltà di Ingegneria. Online at: http://www.uniroma2.it/didattica/mss1/ deposito/Introduzione _alla_simulazione.pdf.

[58] Kettinger W. J, Teng J. T, Guha S, (1997) Business process change: a study of methodologies, techniques, and tools," MIS quarterly, pp. 55–80.

[59] Kiran M. (2014).Using FLAME Toolkit for Agent-Based Simulation: Case Study Sugarscape Model.

[60] Kroes, N., 2013. "Open Access to science and data = cash and economic bonanza". Speech held at the Open Access conference, Berlin.

[61] Law W M. David Kelton D W. SIMULATION MODELING AND ANALYSIS.

[62] Lawrence Edward Henesey. (2006) Multi-Agent Systems for Container Terminal Management.

[63] LeBaron and Tesfatsion, 2008; Farmer and Foley, 2009; Battiston et al., 2016; Turrell, 2016 Schelling, 1969, 1971; Epstein and Axtell, 1996; Axelrod, 1997

[64] Marks R.E. (2007). Validating simulation models: a general framework and four applied examples. Comput. Econ. 30,265.

[65] Malinova M, Hribar B, Mendling J. (2014). A framework for assessing bpm success," Proceedings of the European Conference on Information Systems (ECIS), pp. 1–15.

[66] Overview and User Manual of FLAME. URL: http://flame.ac.uk/

[67] Oracle VM Virtual Box Website. https://www.virtualbox.org/.

[68] Parunak HVD. (1999). Industrial and practical applications of DAI. In: Weiss G, Editor, Multiagent Systems-A modern approach to distributed Artificial Intelligence. MIT Press, Cambridge MA, pp 79-120.

[69] Ponta L, Carbone A, and M. Gilli M. (2011). Resistive transition in disordered superconductors with varying intergrain coupling. Superconductor Science & Technology, vol. 24, no. 1.

[70] Ponta L, M. Raberto M, and S. Cincotti S. (2011). A multi-assets artificial stock market with zero-intelligence traders," Europhys. Lett., vol. 93, p. 28002.

[71] Ponta L, M. Raberto M, and S. Cincotti S. (2011). Information-based multi-assets artificial stock market with heterogeneous agents," Nonlinear Anal. Real World Appl., vol. 12, pp. 1235–1242.

[72] Ponta L, Raberto M, Teglio A, and Cincotti S. (2018). An agent-based stockflow consistent model of the sustainable transition in the energy sector," Ecological Economics, vol. 145, pp. 274–300.

[73] Ponta L, Cincotti S. (2018) Traders networks of interactions and structural properties of financial markets: An agent-based approach. Complexity, vol. 2018.

[74] Ponta L, Pastore S, and Cincotti S. (2018) Static and dynamic factors in an information-based multi-asset artificial stock market," Physica A: Statistical Mechanics and its Applications, vol. 492, pp. 814–823.

[75] Porter B, Lifschitz V and F. van Harmelen. (2007). Handbook of Knowledge Representation. Elsevier B.V. & Wikipedia, Sistema multiagente.

[76] Razavian M, Khosravi R. (2008). Modeling variability in business process models using uml. Information Technology: New Generations, 2008. ITNG 2008. Fifth International Conference. IEEE, pp. 82–87.

[77] Ramollari E, Dranidis D, Simons A. J. (2007). A survey of service oriented development methodologies," The 2nd European Young Researchers Workshop on Service Oriented Computing, vol. 75.

[78] Scientific Data Website. http://www.nature.com/scientificdata/.

[79] Senturk O, Tala N, Ponta L, Cincotti S. (2018). Agent-based model and simulations of the management of ports: the import processes at the port of Genoa. COMPENG 2018.

[80] SliTaz Distribution Website. http://www.slitaz.org/en/.

[81] Simulation. Lecture Notes. Online at: http://www.dis.uniroma1.it/~roma/didattica/SSS09-10/parteH.pdf

[82] Stodden, V. C, (2010). Reproducible research: Addressing the need for data and code sharing in computational science. Computing in Science & Engineering 12 (5), 8–12.

[83] “The EURACE Project.” (Apr 12, 2013) http://www.eurace.org.

[84] Tumer K and Agogino A. (2009). Improving air traffic management with a learning multiagent system. IEEE Intelligent Systems, vol. 24, no. 1, pp. 18–21.

[85] Weiss G. (1999). Multiagent Systems, a modern approach to distributed artificial intelligence. MIT Press, Cambridge.

[86] Wikipedia, XML. URL: https://it.wikipedia.org/wiki/XML

[87] Virtual Appliance. http://www.wiwi.unibielefeld.de/lehrbereiche/vwl/etace/Eurace_Unibi/id/Virtual_Appliance.

[88] Wohed P, W. M. van der Aalst, Dumas M, A. H. ter Hofstede, Russell N. (2006). On the suitability of bpmn for business process modelling. Business Process Management, vol. 4102, pp. 161–176.

[89] Wooldridge M, Jennings N. (2002). An introduction to Multiagent Systems. John Wiley and sons. New York.

[90] Wooldridge M, Wiebe van der Hoek. Multi-Agent Systems.

[91] Wikipedia, Port. Online at: https://en.wikipedia.org/wiki/Port

[92] Wikipedia, Port of Genova. URL: https://it.wikipedia.org/wiki/Port_of_Genova

[93] Varga A. (2005).Discrete Event Simulation System, Version 3.2, User Manual.

[94] Vom Brocke J, Rosemann M. et al. (2010) Handbook on business process management. Springer.

[95] Young Yun W, Choi S. Y. (1999). A simulation model for container-terminal operation analysis using an object-oriented approach, Int. J. Production Economics 59 221—230.

Abbreviations

ABM	Agent-Based Modelling
ABMSs	Agent-based models and simulations
AM	Agent Maritime
API	Application Programming Interface
A3	Customs Entry
BDI	Belief-Desire-Intention
BPMN	Business Process Model and Notation
CC	Clear Code
CCA	Customs Clearance Agent
CD	Customs Declaration
D	Deliverer
DO	Delivery Order
DOC	Leaving documentation
DS	Delivery Service
DSGE	Dynamic Stochastic General Equilibrium
ENS	Entry Summary Declaration
EQ	Equal to
FIFO	First In – First Out
FLAME	Flexible Large-scale Agent Modelling Environment
FP	Finance Police
FSM	Finite State Machines
GASM	Genova Artificial Stock Market
GEQ	Greater than or equal to
GT	greater than
GUI	Graphical User Interface
HPC	High Performance computers
IDE	Integrated Development Environment
LEQ	Less than or equal to
LHS	Left hand side
LT	Less than
MAS	Multi-agent system
MMA	Manifesto Merci in Arrivo
MRN	Movement Reference Number
NEQ	Not equal to
OP	Operator of comparison
OT	Order of Transport
RHS	Right hand side
SC	Security Clearance
SECH	Southern European Container Hub
STEM	Science, Technology, Engineering and Mathematics
TEU	Twenty-foot equivalent unit
UML	Unified Modelling Language
XML	Extensible Markup Language
VTE	Voltri Terminal Europa
VM	Virtual Machines