

Old Dominion University

## ODU Digital Commons

---

Electrical & Computer Engineering Theses &  
Dissertations

Electrical & Computer Engineering

---

Fall 2002

# Object-Oriented Architecture for Concurrent Processes in a Port Simulation

Reejo Mathew  
*Old Dominion University*

Follow this and additional works at: [https://digitalcommons.odu.edu/ece\\_etds](https://digitalcommons.odu.edu/ece_etds)



Part of the [Computational Engineering Commons](#), [Computer Engineering Commons](#), [Computer Sciences Commons](#), [Operational Research Commons](#), and the [Transportation Commons](#)

---

### Recommended Citation

Mathew, Reejo. "Object-Oriented Architecture for Concurrent Processes in a Port Simulation" (2002). Master of Science (MS), Thesis, Electrical & Computer Engineering, Old Dominion University, DOI: 10.25777/crwb-h067  
[https://digitalcommons.odu.edu/ece\\_etds/421](https://digitalcommons.odu.edu/ece_etds/421)

This Thesis is brought to you for free and open access by the Electrical & Computer Engineering at ODU Digital Commons. It has been accepted for inclusion in Electrical & Computer Engineering Theses & Dissertations by an authorized administrator of ODU Digital Commons. For more information, please contact [digitalcommons@odu.edu](mailto:digitalcommons@odu.edu).

**OBJECT-ORIENTED ARCHITECTURE FOR CONCURRENT  
PROCESSES IN A PORT SIMULATION**

by

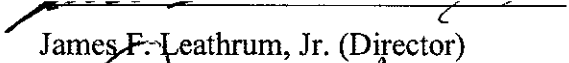
Reejo Mathew  
Bachelor of Engineering, June 2000  
Sardar Patel College of Engineering, Mumbai University

A Thesis submitted to the Faculty of  
Old Dominion University in Partial Fulfillment of the  
Requirement for the Degree of

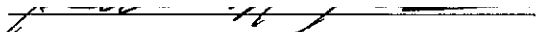
MASTER OF SCIENCE  
COMPUTER ENGINEERING  
OLD DOMINION UNIVERSITY

December 2002

Approved by:

  
James F. Leathrum, Jr. (Director)

  
Roland R. Mielke (Member)

  
Frederic Mckenzie (Member)

# **ABSTRACT**

## **OBJECT-ORIENTED ARCHITECTURE FOR CONCURRENT PROCESSES IN A PORT SIMULATION**

Reejo Mathew  
Old Dominion University, 2002  
Director: Dr. James F. Leathrum

An architectural model to represent the resources and infrastructure within a cargo terminal, as well as to support the concurrent but opposite flow of cargo within the terminal, is presented in this thesis. The model supports the configuration of an individual cargo terminal according to its characteristics, as well as the processing involved in the simultaneous flow of cargo in opposite directions through the terminal. This is useful in the analysis of the flow of military cargo and aids the decision-making process to increase the efficiency and throughput of any military operation. This model has been developed with the aim of supporting the simulation and analysis of the flow of at least 200,000 pieces of cargo with the cargo being modeled as individual pieces. It is also envisioned that the model be used to connect multiple cargo terminals to represent an end-to-end concurrent cargo flow from a cargo depot within the Continental United States to a Theater of War.

Co-Directors of Advisory Committee:	Dr. Roland R. Mielke
	Dr. Frederic Mckenzie

This thesis is dedicated to all the people  
who made a difference in my life.

## ACKNOWLEDGMENTS

I would like to extend special thanks to my thesis advisor, Dr. James Leathrum, Jr. for the knowledge and guidance that he has given me during the two years that I have worked with him. I would also like to thank the co-directors of my advisory committee, Dr. Roland Mielke and Dr. Rick Mckenzie, for all their help and support.

I would like to thank Mr. Joseph Joines at the Military Traffic Management Command Transportation Engineering Agency (MTMCTEA) for data and clarifications during the model development process. I would also like to thank my colleagues at the Virginia Modeling, Analysis, and Simulation Center (VMASC) and, in particular, Mr. Taylor Frith for his input during the model development as well as its documentation in Microsoft VISIO.

Last but not least, I would like to thank my family and friends for being there for me all my life.

# TABLE OF CONTENTS

	Page
LIST OF TABLES .....	vii
LIST OF FIGURES .....	viii
LIST OF GRAPHS .....	xii
ABBREVIATIONS AND ACRONYMS .....	xiii

## Chapter

I. INTRODUCTION.....	1
1.1.Overview .....	1
1.2.Problem Statement .....	1
1.3.Benefits of Simulation .....	2
1.4.Requirements .....	3
1.5.Architectural Approach .....	4
1.6.Chapter Overview .....	5
II. BACKGROUND .....	6
2.1 Overview .....	6
2.2 Existing Port Models .....	6
2.3 CPORTS .....	9
III. MODEL DESCRIPTION .....	13
3.1 Overview.....	13
3.2 Problem Statement .....	13
3.3 Model Requirements .....	14
3.4 Architectural Model Description .....	16
3.4.1 Transit Interface .....	17
3.4.2 Cargo Terminals.....	18
3.4.3 Cargo Terminal Areas .....	23
3.5 Interconnection between components .....	29
3.6 Resource Handling for Concurrent Operations .....	31
IV. CONCURRENT RAIL OPERATIONS .....	33
4.1 Overview.....	33
4.2 Description of Rail Operations .....	33
4.2.1 Operational Infrastructure.....	33
4.2.2 Port of Debarkation (POD) Operations.....	34
4.2.3 Port of Embarkation (POE) Operations .....	37
4.2.4 Resource and Infrastructure Contention Between POD and POE operations .....	37

4.3 Simulation of Concurrent Rail Operations .....	42
4.3.1 Concurrent Rail Operations .....	42
4.3.2 Resolving Resource and Infrastructure Contention .....	45
4.4 Analysis and Results .....	47
V. CONCLUSION .....	54
5.1 Achievements.....	54
5.2 Enhancements .....	55
REFERENCES.....	57
APPENDICES	
A. POD Rail Processes .....	59
A.1 Selecting cargo to fill a single train .....	59
A.2 Triggering a train to enter the port .....	65
A.3 Loading a train with its cargo .....	70
A.4 Train leaving the port with its cargo .....	74
B. POE Rail Processes .....	76
B.1 Train with cargo arriving at the port .....	77
B.2 Train entering the port .....	78
B.3 Unloading the train .....	78
B.4 Empty train leaving the port .....	79
VITA .....	84

## LIST OF TABLES

Table	Page
1. Execution times for the simulation .....	53

## LIST OF FIGURES

Figure	Page
1. Architecture and Cargo Processing Models .....	5
2. CPORTS POD Macro Cargo Flow .....	11
3. Cargo Flow from Installation to Theatre of War .....	14
4. <i>Mover, Resource and Transport</i> Classes .....	16
5. <i>TransitInterface</i> Class .....	17
6. Network of Cargo Terminals .....	18
7. <i>CargoTerminal</i> Class .....	19
8. <i>Port</i> Class .....	20
9. A Complete Port .....	21
10. Port and External Transportation Interface .....	21
11. Port Entry and Exit .....	22
12. Port Level Architecture .....	23
13. <i>CargoTerminalArea</i> Class .....	24
14. <i>PortArea</i> Class .....	25
15. Port Areas .....	26
16. A Complete Port Area .....	27
17. Port Area Entry and Exit .....	27
18. Port Area Level Architecture .....	28
19. Complete Architecture for Concurrent POD / POE Operations .....	30
20. Interconnection between Components .....	29
21. Flow of POD Rail Cargo through a Port .....	35

22. POD Rail Cargo Processes .....	36
23. Flow of POE Rail Cargo through a Port .....	38
24. POE Rail Cargo Processes .....	39
25. Micro-Level Interconnection between Port Areas for Concurrent POD/POE Rail Operations .....	43
26. Macro-Level Interconnection for Concurrent POD/POE Rail Operations .....	44
27. POD Scenario File Routes for Port Ad Dammam .....	48
28. POE Scenario File Routes for Port Ad Dammam .....	49
29. Model Flow Diagram for POD <i>TransportArriveRail</i> process in a <i>Port</i> Object .....	59
30. <i>TrainStatus</i> Object .....	60
31. Model Flow Diagram for POD <i>InitTrainCargo</i> process in a <i>StagingAreaSet</i> Object .....	61
32. Model Flow Diagram for POD <i>FillRailCargo</i> process in a <i>StagingAreaSet</i> Object .....	62
33. Model Flow Diagram for POD <i>PutCargoOnTrain</i> process in a <i>StagingAreaSet</i> Object .....	63
34. Model Flow Diagram for POD <i>ProcessRailCargo</i> process in a <i>StagingAreaSet</i> Object .....	64
35. Model Flow Diagram for POD <i>GetTrain</i> process in a <i>InterchangeYardAreaSet</i> Object .....	65
36. Model Flow Diagram for POD <i>SelectIYA</i> process in a <i>InterchangeYardAreaSet</i> Object .....	66
37. <i>InterchangeYardArea</i> Object .....	66
38. Model Flow Diagram for POD <i>ResourceArrive</i> process in a <i>InterchangeYardArea</i> Object .....	67

39. Model Flow Diagram for POD <i>SendTrain</i> process in a <i>InterchangeYardArea</i> Object .....	68
40. Model Flow Diagram for POD <i>SelectRSA</i> process in a <i>RailSpurAreaSet</i> Object .....	69
41. Model Flow Diagram for POD <i>ResourceLeave</i> process in a <i>InterchangeYardArea</i> Object .....	69
42. Model Flow Diagram for POD <i>GetRailCargo</i> process in a <i>StagingAreaSet</i> Object .....	70
43. Model Flow Diagram for POD <i>SendRailCargo</i> process in a <i>StagingAreaSet</i> Object .....	71
44. <i>RailSpurArea</i> Object.....	71
45. Model Flow Diagram for POD <i>CargoArrive</i> process in a <i>RailSpurArea</i> Object .....	72
46. Model Flow Diagram for POD <i>ResourceArrive</i> process in a <i>RailSpurArea</i> Object .....	72
47. Model Flow Diagram for POD <i>LoadCargo</i> process in a <i>RailSpurArea</i> Object .....	73
48. Model Flow Diagram for POD <i>CargoLeave</i> process in a <i>RailSpurArea</i> Object .....	74
49. Model Flow Diagram for POD <i>CargoArrive</i> process in a <i>InterchangeYardArea</i> Object .....	75
50. Model Flow Diagram for POD <i>CargoLeave</i> process in a <i>InterchangeYardArea</i> Object .....	76
51. Model Flow Diagram for POD <i>TransportLeaveRail</i> process in a <i>Port</i> Object .....	76
52. Model Flow Diagram for POE <i>TransportArriveRail</i> process in a <i>Port</i> Object .....	77
53. Model Flow Diagram for POE <i>CargoArrive</i> process in a <i>InterchangeYardArea</i> Object .....	78
54. Model Flow Diagram for POE <i>CargoLeave</i> process in a <i>InterchangeYardArea</i> Object .....	79

55. Model Flow Diagram for POE <i>CargoArrive</i> process in a <i>RailSpurArea</i> Object .....	80
56. Model Flow Diagram for POE <i>CargoLeave</i> process in a <i>RailSpurArea</i> Object .....	81
57. Model Flow Diagram for POE <i>ResourceLeave</i> process in a <i>RailSpurArea</i> Object .....	81
58. Model Flow Diagram for POE <i>ResourceArrive</i> process in a <i>InterchangeYardArea</i> Object .....	82
59. Model Flow Diagram for POE <i>ResourceLeave</i> process in a <i>InterchangeYardArea</i> Object .....	83
60. Model Flow Diagram for POE <i>TransportLeaveRail</i> process in a <i>Port</i> Object .....	82

## LIST OF GRAPHS

Graph	Page
1. Closure Times for Port Ad Dammam for various cargo flows .....	50
2. Closure Times for various cargo flows with varying number of Interchange Yard Areas .....	51
3. Closure Times for various cargo flows with varying number of Rail Spur Areas .....	52

## ABBREVIATIONS AND ACRONYMS

ANL	Argonne National Laboratory
CCA	Convoy Construction Area
CONUS	Continental United States
CPORTS	Configurable Port Simulation
IYA	Interchange Yard Area
MTMCTEA	Military Traffic Management Command Transportation Engineering Agency
OCONUS	Outside Continental United States
POD	Port of Debarkation
POE	Port of Embarkation
POPS	Port Operational Performance Simulator
PORTSIM	Port Simulation
RHCT	Riga Harbour Container Terminal
RSA	Rail Spur Area
SA	Staging Area
STON	Short Ton
VMASC	Virginia Modeling, Analysis, and Simulation Center

## *Chapter I*

# INTRODUCTION

## 1.1 Overview

<sup>1</sup>An architectural model to represent the resources and infrastructure within a cargo terminal, as well as to support the concurrent but opposite flow of cargo within the terminal, is presented in this thesis. The model supports the configuration of an individual cargo terminal according to its characteristics, as well as the processing involved in the simultaneous flow of cargo in opposite directions through the terminal. This is useful in the analysis of the cargo flow and aids the decision-making process to increase the efficiency and throughput of any military operation.

## 1.2 Problem Statement

*Transportation Logistics Planning* is used extensively to prepare for the movement of military cargo, i.e., troops, equipment, and supplies, from a military installation to the Theater of War. The initial input of troops into a Theater of War is usually by air. After this initial input, the movement of military cargo, both for offensive purposes as well as for sustainment of the existing force, is by sea, usually through a commercial seaport. For a variety of reasons, both political and economic, only a portion of the entire seaport is available for military operations. For any military operation, it is possible that the flow of cargo from the Theatre of War back to the installation will begin before the flow in the opposite direction has ended, with some military units starting to

---

<sup>1</sup> The reference model for this work is "*A Reconfigurable Object Model for Port Operations*" from the Proceedings of the Summer Computer Simulation Conference, SCSC 2000:603~608.

return to the installation while other units are still being deployed. Moreover, after the initial surge of military equipment and personnel into the Theater of War, cargo supplies meant to sustain the existing force constitute a major portion of the cargo flow. This results in a bi-directional flow of cargo and resources, with cargo moving in both directions competing for limited resources and infrastructure within a cargo terminal. Proper resource and infrastructure allocation is important in any transportation logistics operation. It is important to ensure that the flow of cargo in one direction does not unnecessarily delay, or worse, stall the flow of cargo in the opposite direction. Therefore, resource and infrastructure allocation between the concurrent flows of cargo, as well as the resolution of any contention that might arise, is vital.

It is impractical and expensive to conduct actual military exercises to study the effects that various ship arrival profiles and the availability of port infrastructure and resources for the concurrent cargo flow have on the efficiency of the entire operation. Simulation provides a viable alternative to capture the concurrent flow of cargo within a model. It also enables an analysis of the simulation results for maximum efficiency and throughput within the confines of resources and infrastructure availability.

### **1.3 Benefits of Simulation**

*Simulation* is a cost-effective and simplistic method for modeling a complex physical system with a level of detail just enough to adequately represent a real-world system under consideration. The processing involved in the system needs to be modeled with appropriate detail so that the developed model provides an accurate representation of

the system for a detailed analysis. An analysis of the simulation results can then be used to aid in the decision-making to improve the performance and efficiency of the system.

#### **1.4 Requirements**

Most of the architectural models that have been developed previously have been devoted to modeling the flow of cargo in a single direction within a particular cargo terminal. In order to perform a thorough analysis of extended cargo and transportation logistics within the terminal, an architectural model for the cargo terminal should be able to handle the simulation of concurrent cargo flow in opposite directions. The model represents the resources and infrastructure within a cargo terminal and supports the concurrent but opposite flow of cargo within the terminal. The model is intended to be reconfigurable as well as generic; the same model can be used to represent an airport, a cargo depot, or an intermediate staging area in the Theater of War. It is also envisioned that the model be used to connect multiple cargo terminals to represent an end-to-end concurrent cargo flow from a cargo depot to the Theater of War.

Previous models usually took 60 minutes to simulate 8000-10000 pieces of cargo with a similar level of detail. This model has been developed with the aim of supporting the simulation and analysis of the flow of at least 200,000 pieces of cargo with the cargo being modeled as individual pieces. Additionally, the model has been developed to execute within an execution time that would make the model useful in a real-world situation, i.e., crisis management.

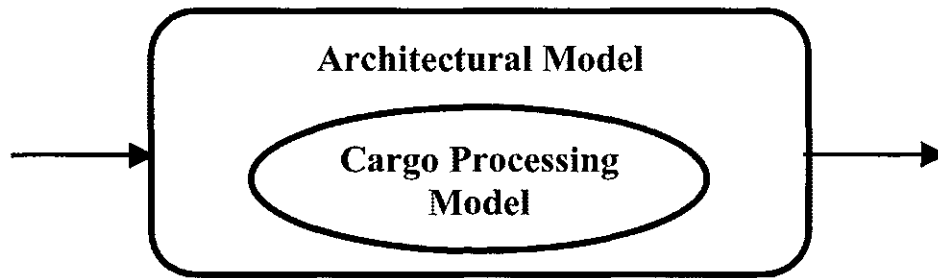
## 1.5 Architectural Approach

*Object-oriented programming* was developed with the basic intention of supporting the simulation of real-world systems. In object-oriented methodology, a simulation program is written to simulate the states and activities of real-world objects. An *attribute* represents the physical properties of the system. A *method* is an operation that can modify the behavior of the object by manipulating its attributes. A real-world system involves multiple entities interacting with each other through the lifetime of the system. Object-oriented methodology aids in simulating this process by instantiating multiple objects and observing the interaction of the objects.

Hence, the model for concurrent process in a port simulation has been developed using an object-oriented, top-down approach because of the following properties:

- **Encapsulation:** The properties and behavior of a physical object is concentrated within a single simulation object.
- **Modularity:** An object is independent of any other object.
- **Inheritance:** This property is vital for a hierarchical structure with multiple objects inheriting the properties of a single object and then personalizing their attributes and behavior.

The architectural model is intended to be a shell around the cargo processing model, which is unique to each cargo terminal as well as to each area within the terminal. The model acts as an interface for the cargo processing model with the outside world. This structure is shown in Figure 1.1.



**Figure 1.1 Architectural and Cargo Processing Models**

## **1.6 Chapter Overview**

Chapter 2 briefly mentions the related research in the field of port simulations. The chapter then describes the CPORTS project. The simulation model, which is the focus of this thesis, has been developed mainly in support of the CPORTS project.

Chapter 3 lists the requirements of the model for concurrent processes in a port simulation. The chapter then provides a complete description of the simulation model.

Chapter 4 briefly describes the various rail operations that occur in a port. The Port of Debarkation (POD) and Port of Embarkation (POE) operations are described separately. A more detailed description is available in Appendices A and B. The chapter then provides a detailed description and analysis of the manner in which the simulation model described in Chapter 3 is used to adequately represent the concurrent POD / POE rail operations in the port.

Chapter 5 concludes the thesis and offers suggestions on improvements and additions to the model.

## *Chapter II*

### **BACKGROUND**

#### **2.1 Overview**

This chapter gives a brief description of the research that has been done by various research groups in developing port simulations for the purpose of transportation logistics planning and analysis, both for military as well as commercial cargo.

#### **2.2 Existing Port Models**

Port simulation models that currently exist in literature differ widely in their objectives, model complexity, level of detail and the factors taken into consideration within the model. This difference is mainly due to the variation in the questions that the particular model attempts to answer as well as the model fidelity.

A port model developed at Haifa University, Israel, is described in [3]. It models different types of cargo arriving on ships and resources used until ships dock at the berth and cargo is unloaded from the ships. The model addresses multiple port functions, ship arrival profiles, cargo composition on ships and availability, allocation, and utilization of resources, as well as the co-ordination between terminals in more than one port. The focus of the model, however, is the idle times for ships waiting to dock at the port.

A model aimed at improving logistic processes at the Riga Harbour Container Terminal (RHCT) is discussed in [4]. It models the RHCT in detail with the arrival of

ships, unloading of containers, and crane capabilities. It also models the storage of containers at import yards and their transfer to export yards to be loaded onto trains to be carried out of the cargo terminal. The model is basically a queuing network model with the explicit intent of modeling the RHCT, which leads to a rigid structure for the model.

A decision support system for an intermodal container terminal at La Spezia Cargo Terminal in Italy is studied in [5]. The simulation of the terminal is a discrete-event simulation based on the process-oriented paradigm. The model captures an import flow, i.e., containers arriving by ships to be transported to their destination by trucks and trains, as well as an export flow, i.e., containers arriving by trucks and trains and leaving the terminal on ships. It also captures the storage of containers, resource allocation, and scheduling for loading and unloading operations. The model is at a high level of abstraction focusing mainly on resource allocation between both flows of cargo, import and export.

A more generic model for a cargo terminal is proposed in [6], [7], with the focus being on resource allocation for container operations within the terminal. The logistics planning involved in the transport of containers and modeling of operations until ships dock at the port for unloading are discussed in [6]. The model aims to reduce the ship turnaround time (time a ship has to spend at a terminal) by optimally utilizing the terminal resources. Genetic algorithms are used in [7] to create ship arrival profiles that improve the ship turnaround time as well as to aid in the decision-making about resource

allocation and terminal organization (container layout within the terminal). Neither model, however, models cargo handling.

All the cases mentioned above are limited by the fact that they model commercial operations at a specific cargo terminal with a varying level of detail about the operational behavior of the terminal. Most of the above research put forward the argument that the model can be extended to include an improved level of detail.

Transportation logistics planning for military operations has also been previously modeled. PORTSIM (Port Simulation) [8], [9], developed by Argonne National Laboratory (ANL), addresses two modes of military cargo operations: Port of Embarkation (POE) and the Port of Debarkation (POD). The POE mode deals with the arrival of cargo at the port via trains and highway transports, staging the cargo, and loading a ship with cargo. The POD mode focuses on the activities of unloading cargo from a ship, staging, parking, and inspection of the cargo, and clearing the cargo from the port using trains and highway transports. PORTSIM has a fully functional POE model; however, the POD model has not been correctly and fully developed. Moreover, PORTSIM suffers from the following problems: mutually exclusive POE and POD processes, fixed cargo flow, unmanageable code, slow execution times, and single-run capability only.

POPS (Port Operational Performance Simulator) [11], developed by the Military Traffic Management Command Transportation Engineering Agency (MTMCTEA),

estimates the capability of a seaport to handle military cargo. POPS analyzes cargo at the aggregate (Short Ton (STON)) level rather than at the entity level. It is a statistical analysis of the ability of the port rather than a true port model.

## 2.3 CPORTS

CPORTS (Configurable Port Simulation) [1], [2] is a discrete-event simulation developed by Dr. James F. Leathrum, Jr. and his research group at Virginia Modeling, Analysis, and Simulation Center (VMASC) for MTMCTEA. CPORTS is used to analyze the movement of military cargo through a seaport.

The seaports (henceforth referred to as ports) under consideration are commercial ports, and therefore, complete access to all port facilities is generally not available. CPORTS is useful for comparing and selecting ports as well as in determining port throughput capability, and allocation and utilization of critical resources. The architecture for concurrent POD / POE processes in a port simulation, the focus of this thesis, has been developed mainly in support of the CPORTS project as well as to support the networking of cargo terminals for a complete intra-theater analysis.

CPORTS is designed to obtain the following data:

- Port Closure Time.
- Port Throughput Capability.
- Cargo Data (during its flow through the port) for detailed analysis.
- Ship Data.

- Stranded Cargo Data.

CPORTS is also used to identify the following:

- Potential bottlenecks within the port.
- Port resources limiting movement of cargo.
- Implications if certain port infrastructure or resources are constrained or unavailable.
- Impact of various ship arrival profiles.

CPORTS is designed with the intent of configuring a port according to its specific characteristics rather than designing a generic port with a rigid structure. The general processes in a port are modeled and then access is available to the port expert to specify the process times as well as available infrastructure and resources in the port. CPORTS process modeling addresses the POD mode of operation within a port and has been developed entirely at VMASC.

Figure 2.1 shows the typical flow of POD cargo through a port [12], [16]. Cargo is brought into the port by ships, which are processed through the Anchorage Area for an available space at the set of Berth Areas. Cargo is then unloaded from a ship and is transported by berth side resources to the Staging Areas (SAs). In the SAs, cargo is processed differently based on the mode of transport by which the cargo is scheduled to leave the port. Cargo scheduled to leave by highway transports is processed through a Loading Area and leaves through a gate. Cargo scheduled to leave by convoy is

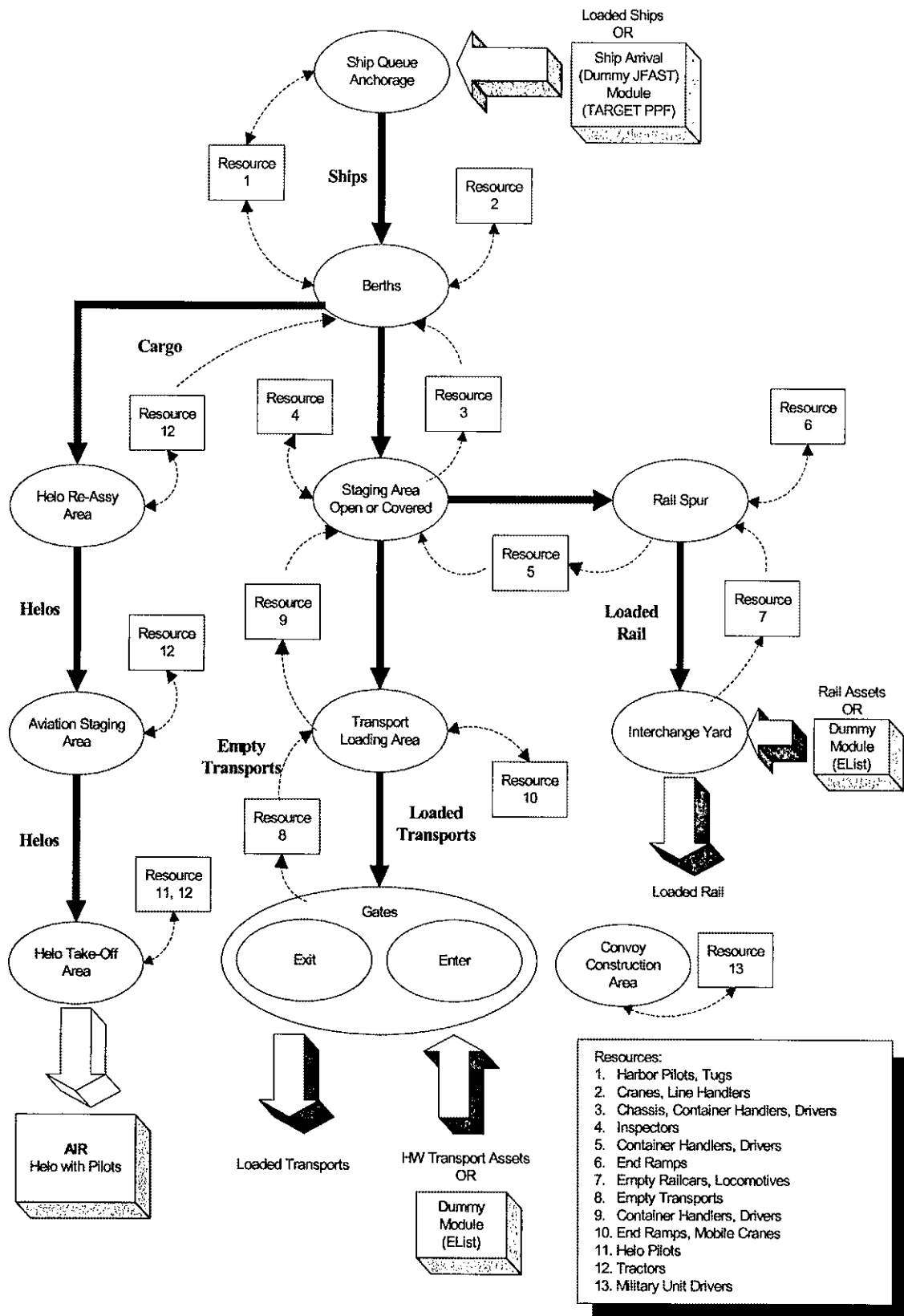


Figure 2.1 CPORTS POD Macro Cargo Flow

assembled in the Convoy Construction Area (CCA). Cargo scheduled to leave by rail is processed through the Rail Spur Area (RSA) and the Interchange Yard Area (IYA) to be loaded onto trains, which carry the cargo out of the port.

## *Chapter III*

### **ARCHITECTURAL MODEL**

#### **3.1 Overview**

An architectural model defines the components of a system, specifies the manner in which they interact with one another, and details the interconnection between these components. A long supply chain exists in the flow of military cargo from an installation in the Continental United States (CONUS) to the Theatre of War. New military units are continually deployed and existing units return to the installation while the military operation is still under way. The model described here is required to simulate and analyze one step in the flow of military cargo from an installation to the Theatre of War, as well as the simultaneous flow of cargo in the reverse direction, from the Theatre of War back to the installation.

#### **3.2 Problem Statement**

The architectural model is designed to support the concurrent, but opposite, flow and processing of cargo through a cargo terminal. It supports the configuration of a cargo terminal as well as the interconnection and intraconnection between cargo terminals to facilitate the physical movement of cargo. A cargo terminal should be able to handle cargo flowing in opposite directions simultaneously as it is conceivable that the flow of cargo from the Theatre of War back to the installation will begin before the flow in the opposite direction has ended with some military units starting to return to the installation while other units are still being deployed. This results in a bi-directional flow of cargo

and resources, with cargo moving in both directions competing for a limited number of resources within a cargo terminal. Figure 3.1 shows the typical two-way flow of cargo between cargo terminals. Cargo is transported from the installation to a CONUS port, from where it is transferred to an Outside Continental United States (OCONUS) port. Cargo is then delivered from the OCONUS port to the Theatre of War. A reverse flow of cargo follows the same logistical path.



**Figure 3.1 Cargo Flow from Installation to Theatre of War**

Most of the architectural models that have been developed previously for this purpose have been devoted to modeling the flow of cargo in a single direction [8], [9], [10], [11], a basic limitation. Cargo flowing in the opposite direction has a direct effect on the availability of resources and infrastructure in a cargo terminal. Cargo has to wait longer for infrastructure in the cargo terminal to be freed up and / or for a suitable resource to become available. This in turn affects the clearance time for the cargo, i.e., the time for the cargo to flow through the cargo terminal. It also affects the order in which cargo is cleared from a cargo terminal depending on the precedence given to cargo flowing in a particular direction. Moreover, improper resource or infrastructure allocation between the concurrent flows of cargo within a cargo terminal can lead to the system being deadlocked. It is possible that both cargo flows might be waiting for resources and infrastructure to be freed up by the other cargo flow leading to a deadlock.

It is therefore prudent and essential to develop a model for a cargo terminal capable of handling concurrent operations. This enables an expert to study and correctly analyze the effect that the cargo flowing in one direction has on the infrastructure and resources of the cargo terminal as well as on the cargo flowing in the opposite direction over the full lifetime of the operation.

### 3.3 Model Requirements

The architectural model has been developed with the intention of satisfying the following basic requirements:

- **Concurrent Port of Debarkation (POD) and Port of Embarkation (POE) operations through the same cargo terminal.**

The model should support concurrent POD and POE operations through a single cargo terminal. This is the focus of this model.

- **Initialization of multiple cargo terminals simultaneously.**

The model should support the initialization of multiple cargo terminals simultaneously as well as the flow of cargo between them. This enables the simulation of an end-to-end cargo flow from point of origin to destination.

- **Reconfigurable Cargo Terminals.**

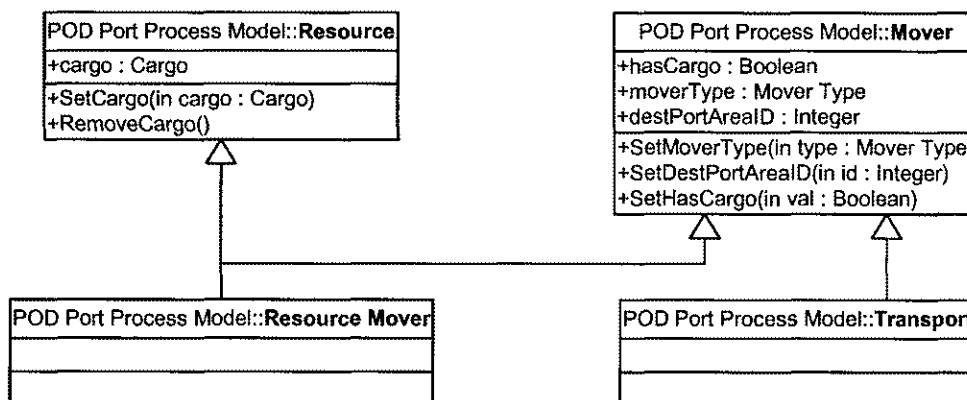
An individual cargo terminal should have the ability to be initialized independent of other cargo terminals. The cargo terminals should also be reconfigurable, that is, the infrastructure and the resources in the cargo terminal as well as the sequence of cargo flow through the areas in a single terminal should be changeable.

- **Simulation of 200,000 pieces of cargo.**

The model should support the simulation of 200,000 pieces of cargo within a reasonable execution time.

### 3.4 Architectural Model Description

The architectural model to support concurrent POD / POE operations is described. The architecture is defined to be object-oriented and is developed using a hierarchical, top-down approach. The purpose of this architecture is to adequately satisfy the model requirements described in the previous section. Moreover, special attention has been given to the flexibility, robustness, scalability, and performance of the model. The architecture is designed to be flexible enough to incorporate changes in the model as well as robust enough to ensure that the changes do not break the architecture. The architecture is also meant to be scalable in terms of new cargo terminals being initialized or new cargo terminal areas being added to a particular cargo terminal. The architecture is built keeping in mind that any changes to the model should not cause any substantial degradation in performance because of the architecture itself.



**Figure 3.2 Mover, Resource, Transport and ResourceMover Classes**

The main purpose of the architecture is to model the flow of cargo. This is achieved by using a *Mover* class. A mover is an object that carries cargo from one physical port area to another. Resource movers and transports that are further used to model the flow of cargo to and from a port area are all derived from the *Mover* class and the *Resource* class. Figure 3.2 shows this class structure.

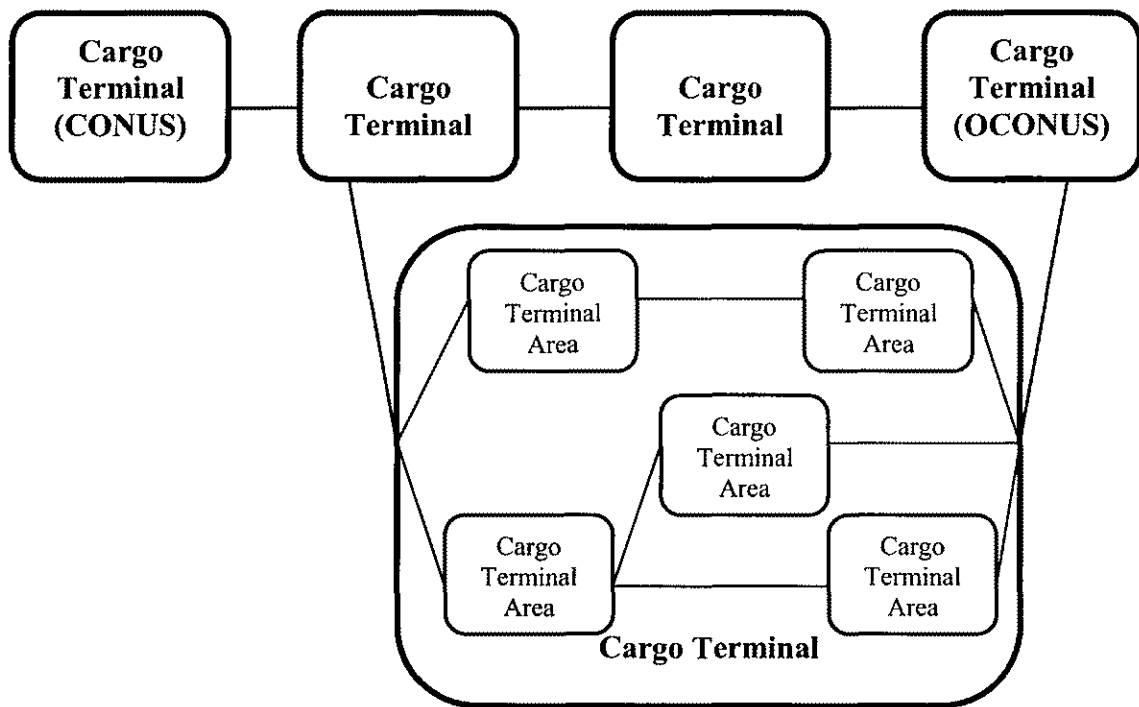
### 3.4.1 Transit Interface

At the top level, the flow of cargo between cargo terminals is modeled. The cargo transportation network is comprised of a set of cargo terminals interconnected by a transportation infrastructure [1]. This transit interconnection interface is modeled using the *TransitInterface* class shown in Figure 3.3.

POD Port Process Model::Transit Interface
-transit : Transit
+Entry Point(in mover : Mover)
+Exit Point(in mover : Mover)
«signal»-Individual Objects override these methods()

**Figure 3.3 *TransitInterface* Class**

The *TransitInterface* is used for a common interface between various objects that are used in the simulation. Each object used in the simulation that represents infrastructure within the cargo terminal inherits the *TransitInterface* object. This enables the modeling of the transit of resources and transports through the cargo terminal with the *ExitPoint* of one connected to the *EntryPoint* of the other. The same *TransitInterface* object can be used to connect cargo terminals to model an end-to-end flow of cargo. The result is a hierarchical structure as shown in Figure 3.4.

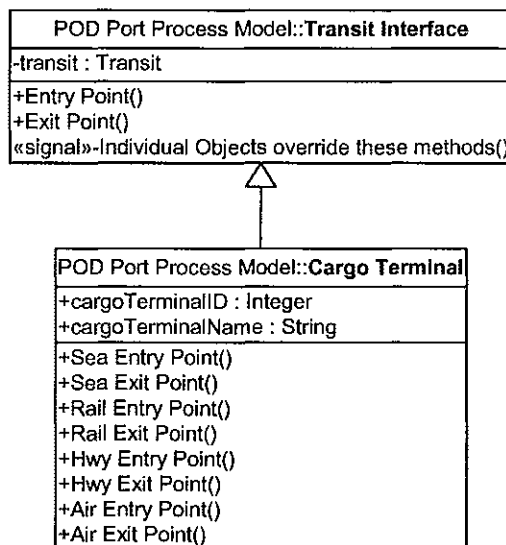


**Figure 3.4 Network of Cargo Terminals**

### 3.4.2 Cargo Terminals

The cargo terminals include the installation (point of cargo origin) and the destination (usually the Theatre of War) as well as intermediate points of storage (CONUS or OCONUS ports) and change of transportation or transfer. The cargo terminals are modeled using the *CargoTerminal* class shown in Figure 3.5.

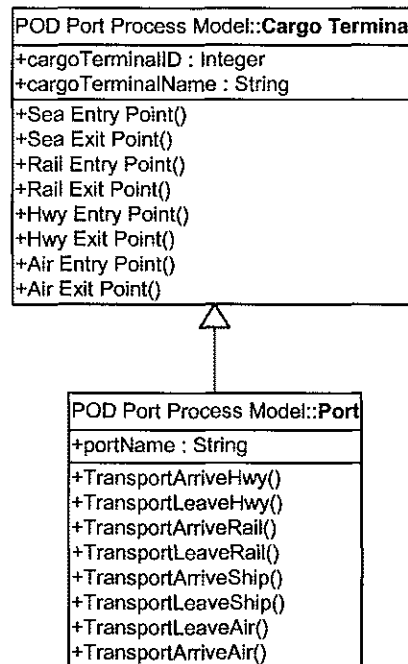
The *CargoTerminal* class encapsulates a single cargo terminal, clearly defining the entry and exit points for that terminal as well as the internal characteristics and operations of the terminal. The main purpose of the *CargoTerminal* class is to provide interfaces between the cargo terminal and the external cargo transport infrastructure [1]. It has individual entry and exit points for the different types of cargo that are handled by



**Figure 3.5 CargoTerminal Class**

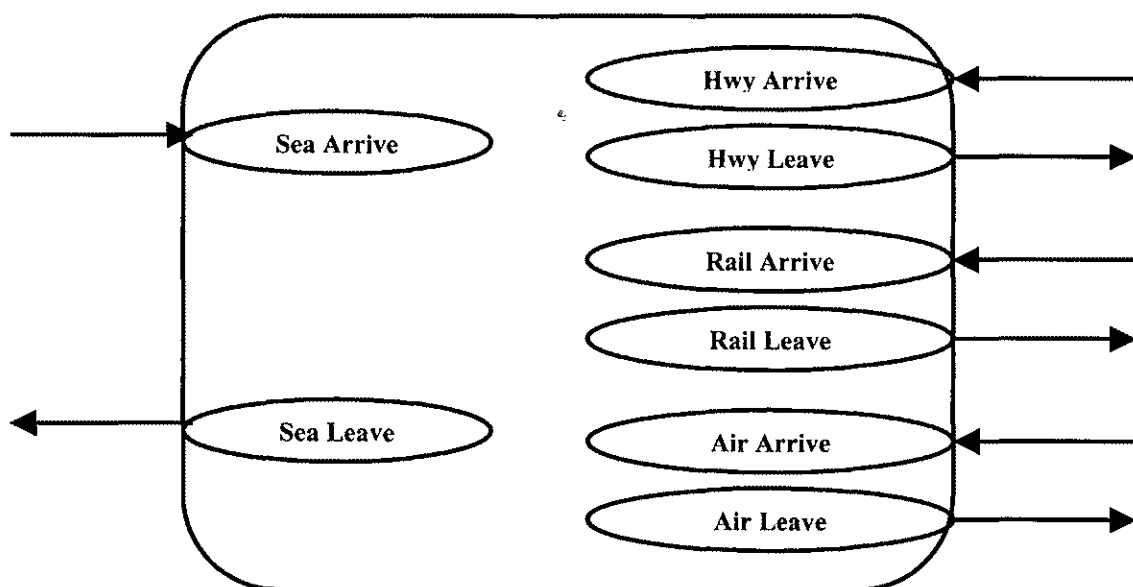
the cargo terminal, namely Sea, Rail, Highway and Air. The *EntryPoints* are called by an external transportation infrastructure object for transports or resources arriving at the terminal. The *ExitPoints* are called by the processing infrastructure within the cargo terminal to pass the transports back to the external transportation infrastructure object when the processing within the cargo terminal is done. Thus, the *CargoTerminal* class provides an interface to the operational model. A seaport and an airport, though varying in their structure and function, can both inherit the *CargoTerminal* class with no changes to its structure. Other cargo terminal examples include installations, warehouses, intermediate storage areas, etc.

The CPORTS project involves the simulation of a port. Figure 3.6 shows a *Port* class. The *Port* class is inherited from the *CargoTerminal* class and encompasses all the infrastructure, operations, and resources within the port. It defines the attributes to



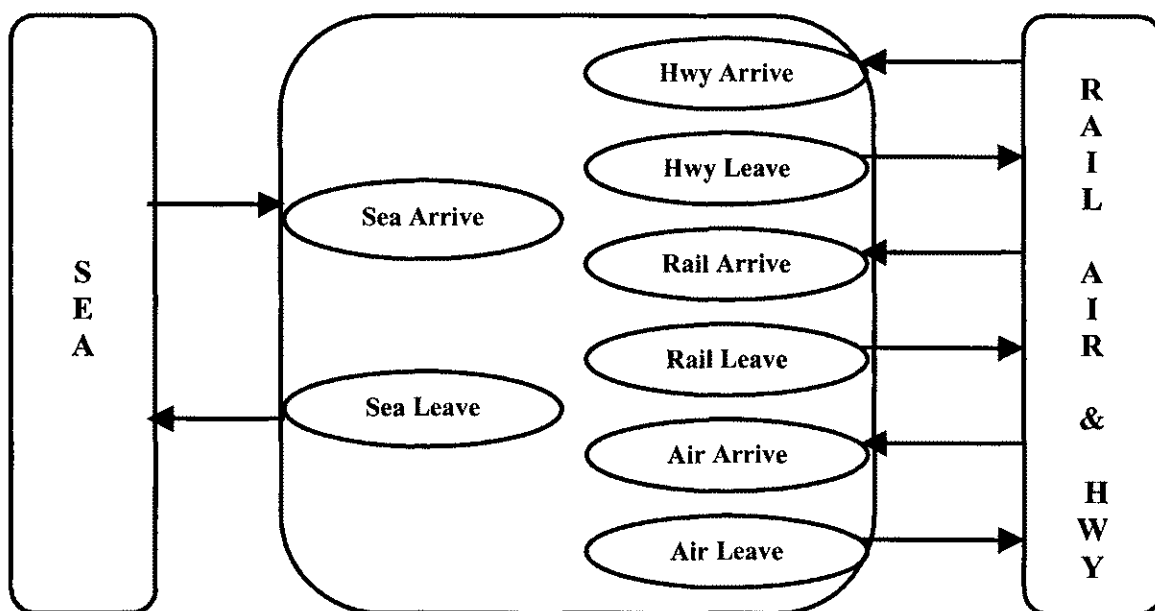
**Figure 3.6 Port Class**

specify the characteristics of the port as well as the infrastructure and resources within the port. It also includes the methods to define the operations within the port. It is completely self-contained for a single port as shown in Figure 3.7. (All objects are represented by rounded boxes in this document, while methods of individual objects are represented by ovals.) It is also desired that the architecture will support initialization and operation of multiple ports concurrently. This can be achieved by instantiating multiple instances of the *Port* object and allowing the *TransitInterface* to provide the means for an interconnection network between the ports. In a stand-alone, single port environment, the *Port* object is connected to the *ShipArrival* module on one end and a *Land / Air Operations* module on the other end. The *ShipArrival* module models the arrival of cargo to a POD port by sea. The *ShipArrival* module feeds ships arriving with cargo to be offloaded into the simulation. The port processes the cargo accordingly. On the other end,



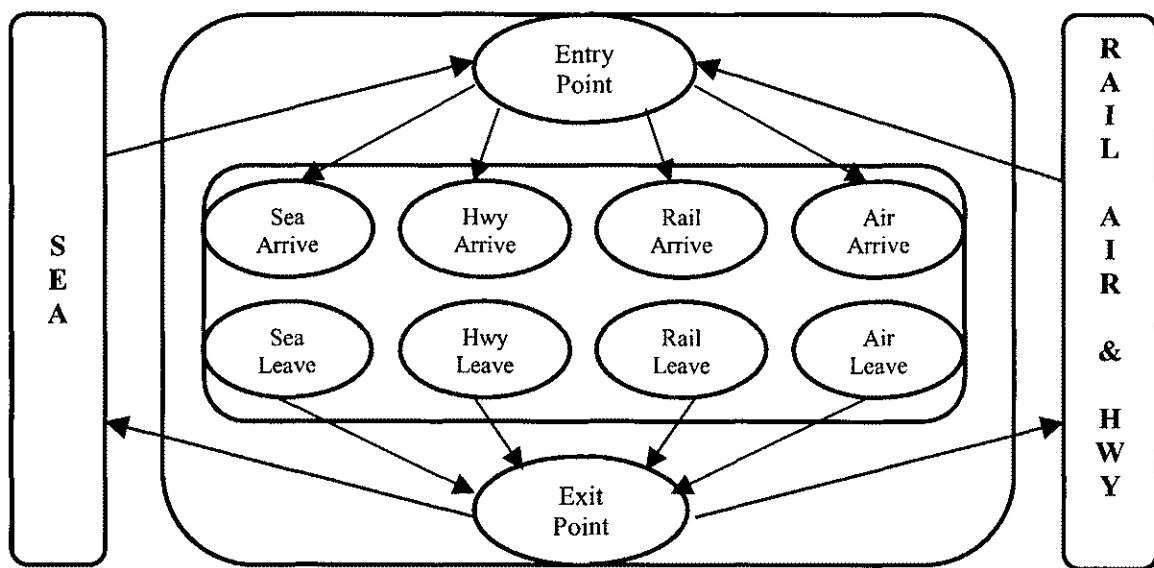
**Figure 3.7 A Complete Port**

the port object is connected to a *Land / Air Operations* module to model the arrival of external transports to transport the cargo out of the port to various destinations. Figure 3.8 shows this interconnection.



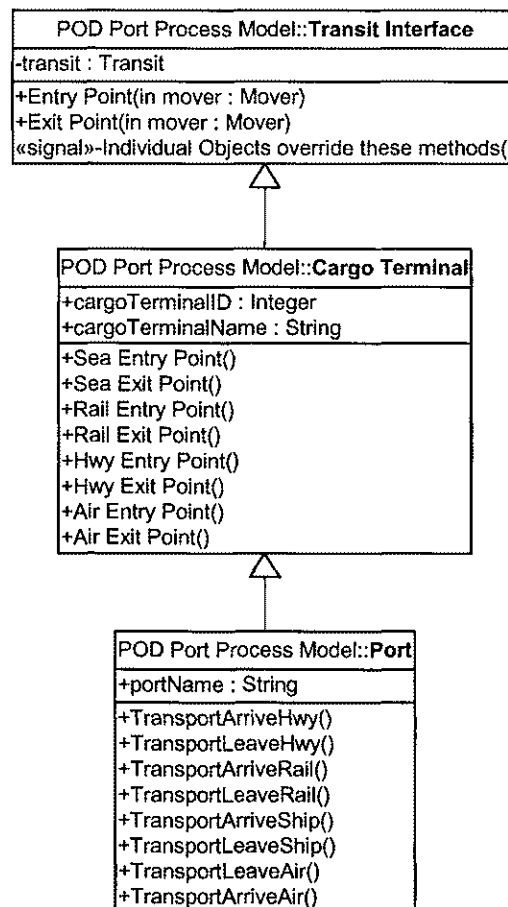
**Figure 3.8 Port and External Transportation Interface**

Figure 3.9 shows the connection of the Air, Highway, Rail and Sea entry and exit points to the *EntryPoint* and *ExitPoint* of the port. This enables the simulation to have a common entry and exit point for all resources and cargo into the port, simplifying the interconnect to external models.



**Figure 3.9 Port Entry and Exit**

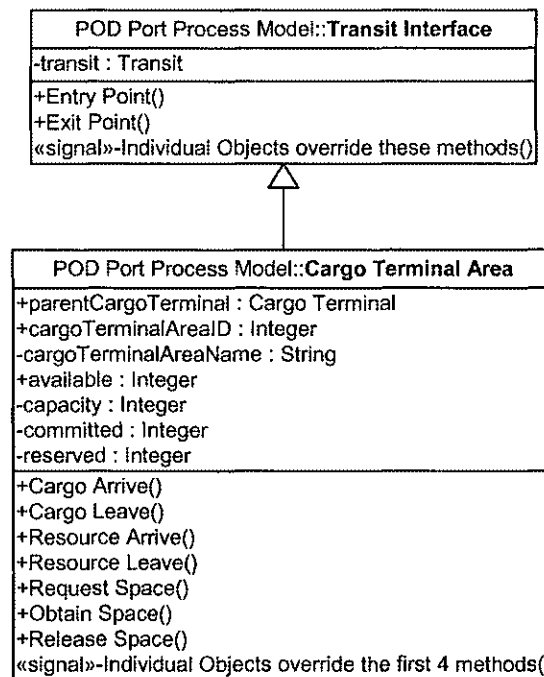
The complete architecture for concurrent POD / POE operations at the port level is shown in Figure 3.10. When a port is initialized, a *Port* object is initialized which inherits all the properties of the *TransitInterface* class and the *CargoTerminal* class. The appropriate methods of the parent objects are overwritten in the *Port* object to reflect the operations and infrastructure of the particular port.



**Figure 3.10 Port Level Architecture**

### 3.4.3 Cargo Terminal Areas

The different areas within a cargo terminal are modeled using the *CargoTerminalArea* class shown in Figure 3.11. The *CargoTerminalArea* class has various attributes to identify itself and to establish its association with a parent cargo terminal. It also has methods to signal the arrival of cargo and resources into the cargo terminal area as well as methods to reserve, allocate, and release space in the cargo terminal area.



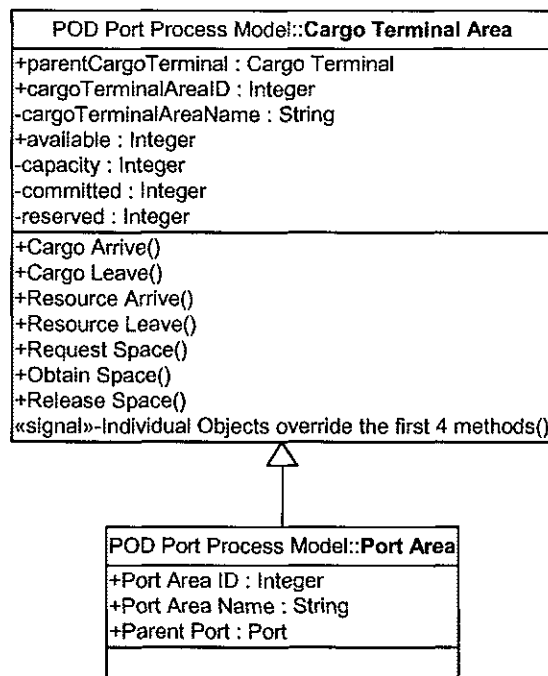
**Figure 3.11 *CargoTerminalArea* Class**

A port area is the physical area within a port where the various port operations are performed. Each port has the following general port areas:

- Anchorage
- Berth Areas
- Staging Areas (SAs)
- Loading Areas
- Convoy Construction Areas (CCAs)
- Gates
- Rail Spur Areas (RSAs)
- Interchange Yard Areas (IYAs)
- Helicopter Re-Assembly Areas

- Helicopter Staging Areas
- Helicopter Takeoff Areas

These port areas are modeled using the *PortArea* class, which is inherited from the *CargoTerminalArea* class. This is shown in Figure 3.12.



**Figure 3.12 *PortArea* Class**

Each individual port may have one or more of these port areas. Certain port areas might be totally absent from a port and need not be initialized at the time of the simulation. All the individual *PortArea* objects inherit the *PortArea* class as well as the methods associated with it. Each individual port area may add various other methods to model the port operation at that particular port area. Figure 3.13 shows the various port areas mentioned earlier.

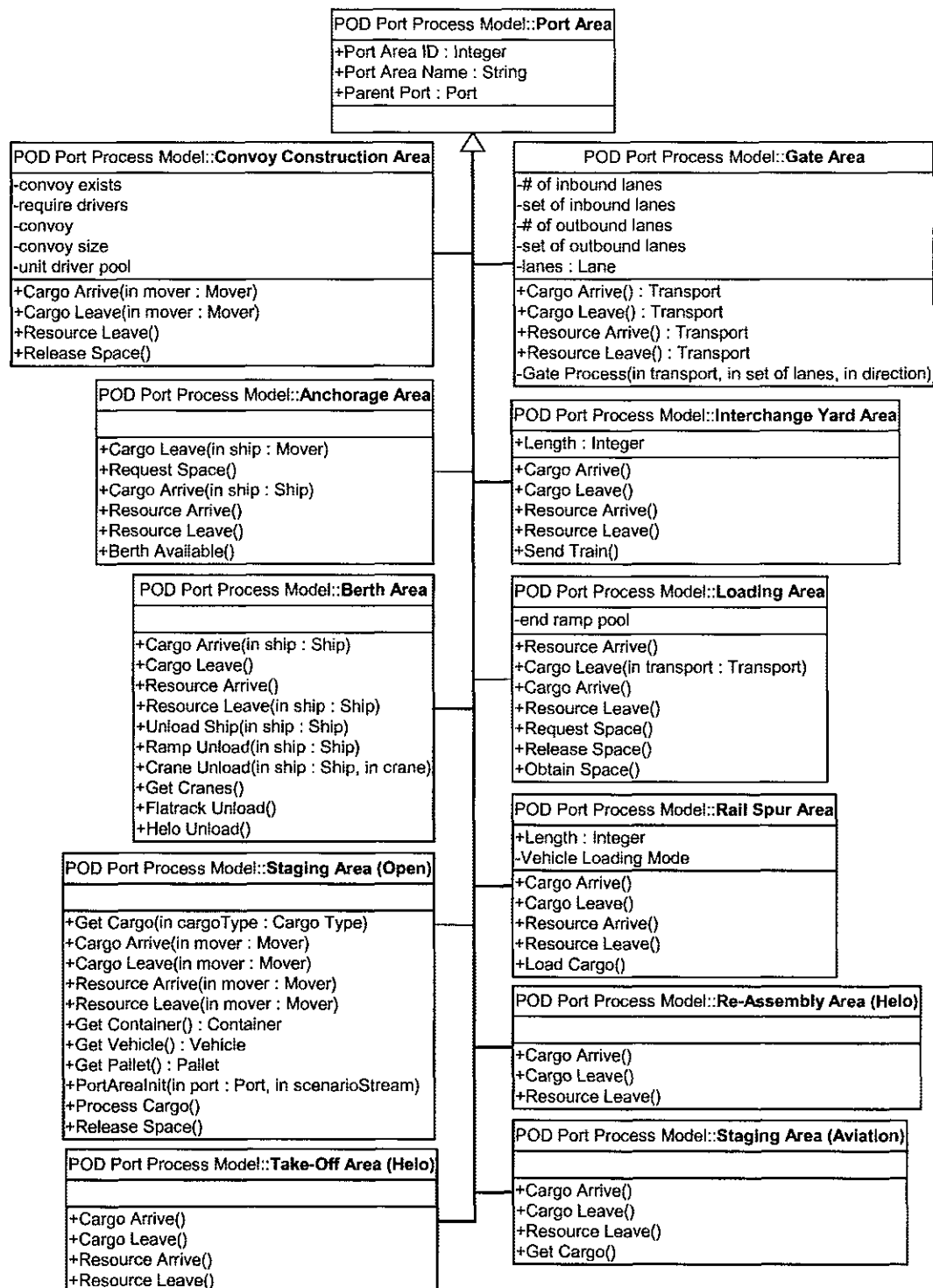
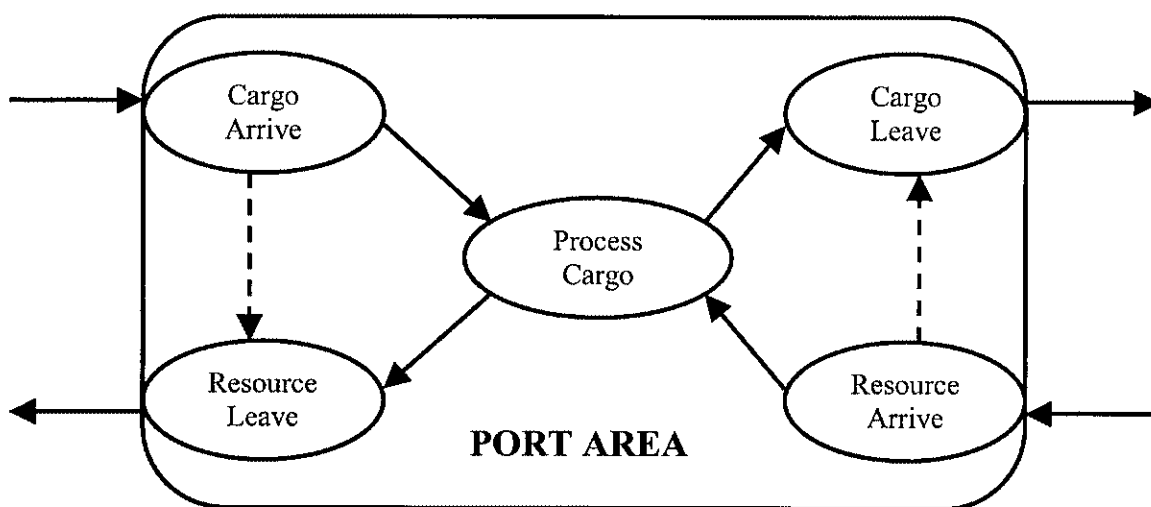
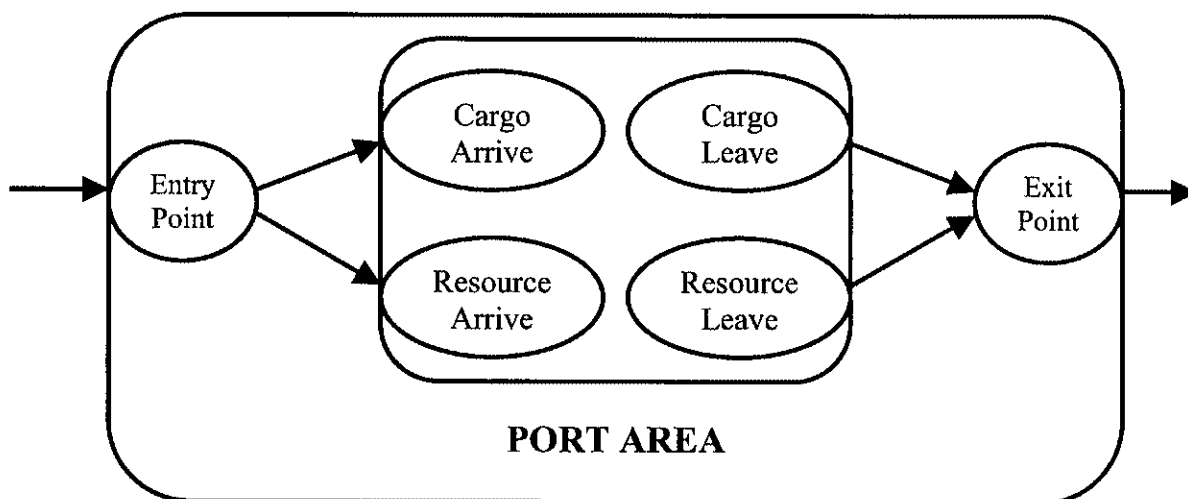


Figure 3.13 Port Areas



**Figure 3.14 A Complete Port Area**

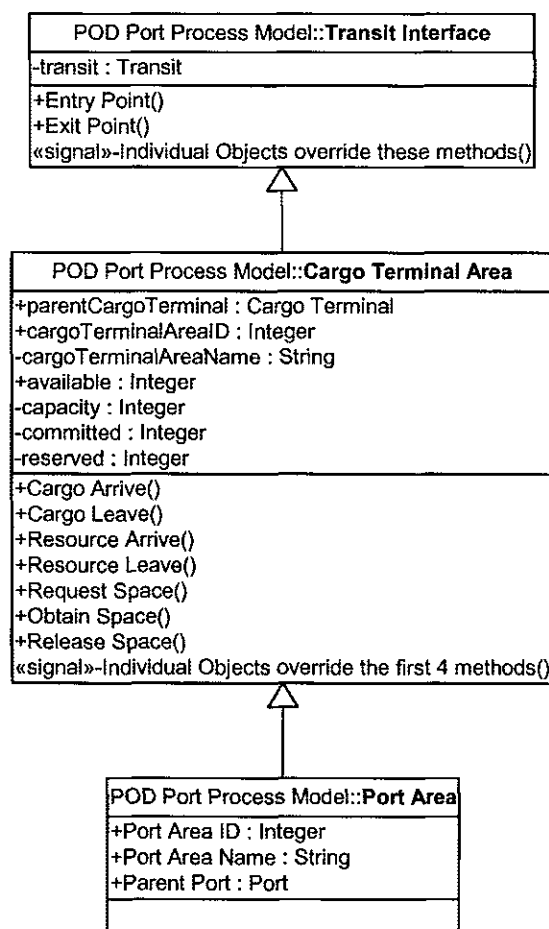
The *PortArea* object is completely self-contained for a single port area as shown in Figure 3.14. Both transports and resources arriving with cargo, as well as arriving to carry cargo out of the port area, enter the area through the *EntryPoint*. A mover (transport or resource) carrying cargo into a port area uses the *CargoArrive* method to enter the port area, while a mover (transport or resource) arriving to carry the cargo out of the port area



**Figure 3.15 Port Area Entry and Exit**

uses the *ResourceArrive* method. The mover, when leaving the port area with cargo, uses the *CargoLeave* method while the empty mover uses the *ResourceLeave* method. After being processed in the port area, transports and resources leave through the *ExitPoint*. The extra wrapping around the port area by the *TransitInterface* provides a single point of entry and exit through the port area. Figure 3.15 shows the entry and exit of all cargo and resources to and from a port area through the *EntryPoint* and *ExitPoint*.

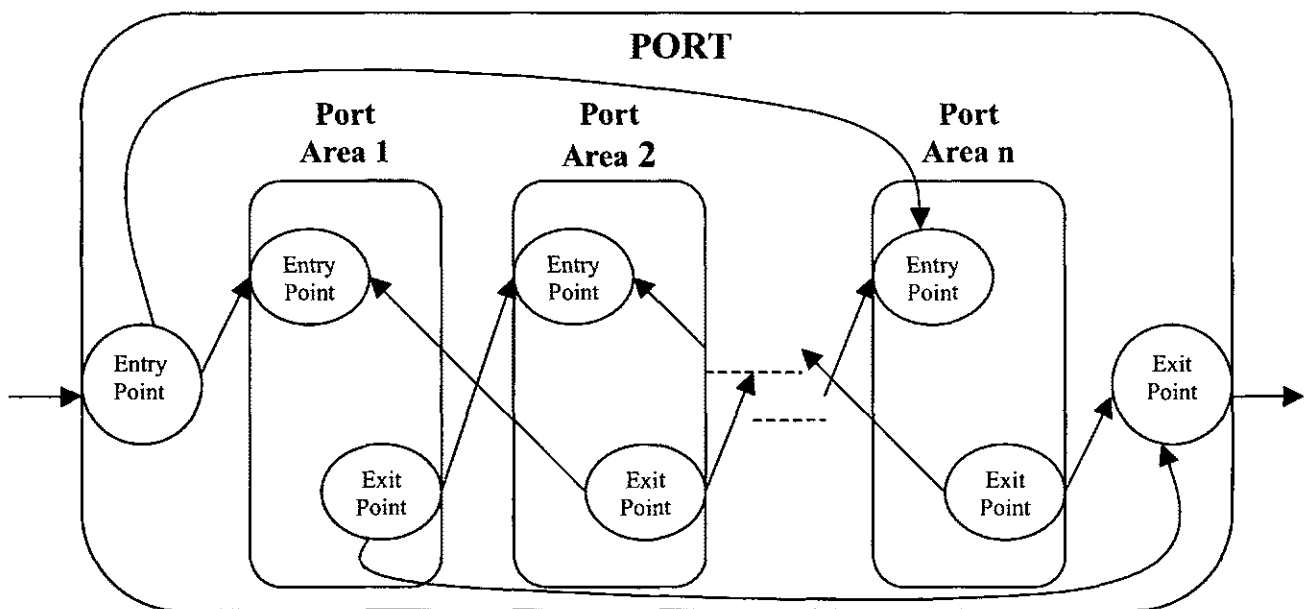
The complete architecture for concurrent POD / POE operations at the port area level is shown in Figure 3.16.



**Figure 3.16 Port Area Level Architecture**

When a port is initialized, a *PortArea* object is initialized for each port area within the port, which inherits all the properties of the *TransitInterface* class and the *CargoTerminalArea* class. The appropriate methods of the parent objects are overwritten in the *PortArea* object to reflect the operations and infrastructure of the particular port area. The complete architecture for concurrent POD / POE operations is shown in Figure 3.17.

### 3.5 Interconnection between Components



**Figure 3.18 Interconnection between Components**

The previous section described the various components of the architectural model. It also described the capabilities provided to connect together multiple cargo terminals as well as the various cargo terminal areas within a single cargo terminal. This section deals with how these capabilities are used to achieve the desired interconnection between components for an end-to-end cargo flow.

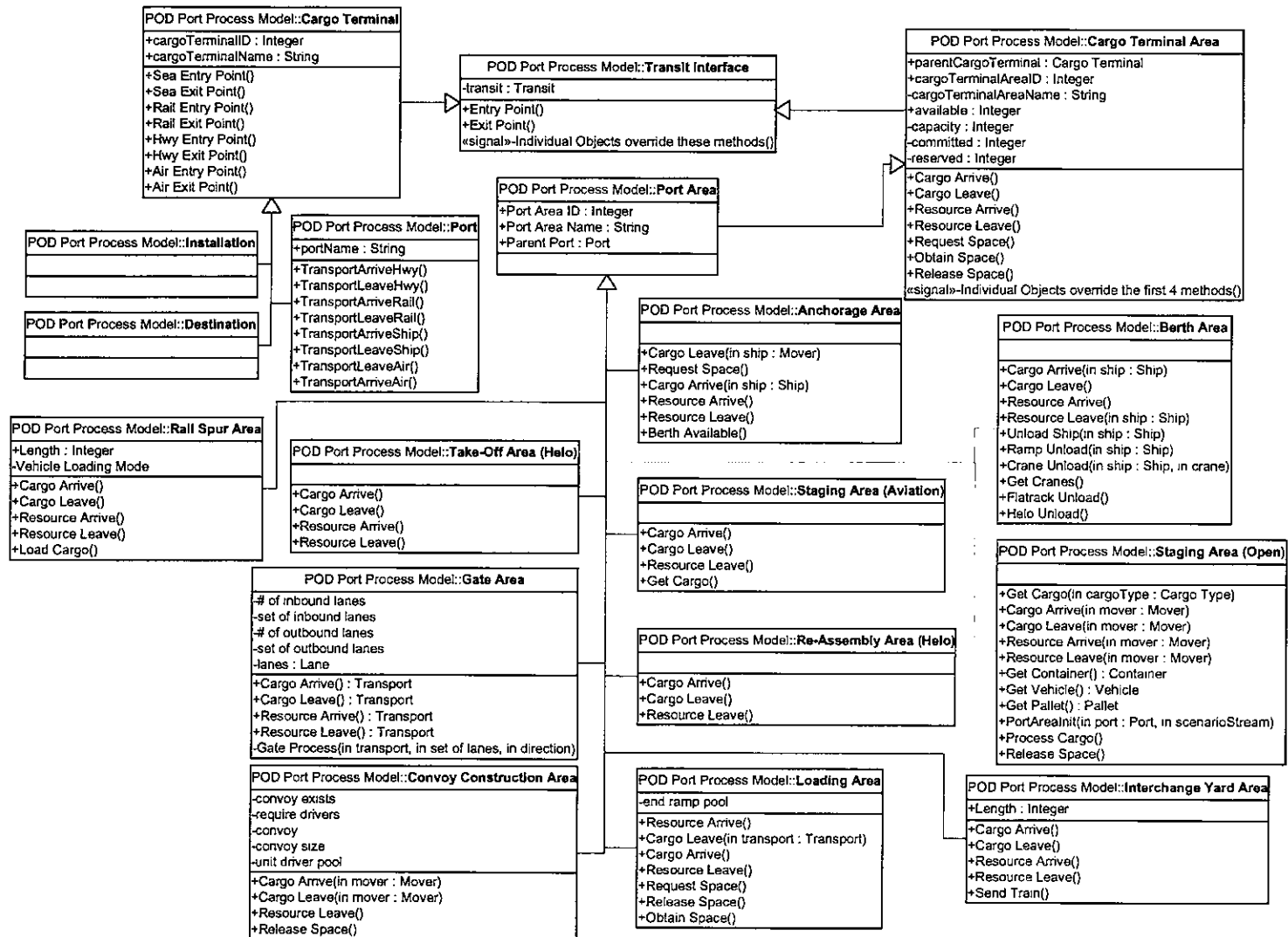


Fig 3.17 Complete Architecture for Concurrent POD / POE Operations

Figure 3.18 shows the manner in which components are interconnected. Cargo is brought into the port by a transport through the *EntryPoint* of the port. Cargo is then processed through the various port areas following a predetermined sequence. This sequence can be different for different types of cargo and is usually dependent on the processing required for a particular type of cargo and the transport that is needed to carry the cargo out of the port. In a particular port area, cargo enters through the *EntryPoint* and is processed accordingly. Cargo then leaves the port area through the *ExitPoint* and is transported to the next port area in its sequence through the *EntryPoint* of that port area.

This interconnection can also be used to support concurrent operations through a single cargo terminal. Both flows of cargo (POD and POE) enter a port area through the *EntryPoint* and are processed based on the mode of cargo. Both flows of cargo then leave the port area through the *ExitPoint* to the next port area in their respective flow. This interconnection can also be extended to connect ports (or cargo terminals) together to achieve an end-to-end flow of cargo from an installation to the Theatre of War and vice versa.

### **3.6 Resource Handling for Concurrent Operations**

Resources are a vital part of the CPORTS simulation and play an important part in the transport of cargo from one port area to another. Resource handling, in turn, becomes very important for concurrent POD / POE operations so that a lack of resources does not hinder the flow of cargo in one or both directions.

There are various approaches to dealing with resources.

- Resources can be placed in a common pool for use by both the POE and POD operations. In this case, the resources are allocated on a first-come-first-served basis.
- Resources can be separated into pools dedicated to either POD or POE operations.
- Resources can be allocated based on a prioritized allocation procedure. The priority can be constant for the entire duration of the simulation or can be changed according to the cargo load flowing in a particular direction at a particular point in the simulation.
- Resources can be placed into pools that are distributed in various port areas and then reallocated based on need.

## *Chapter IV*

### **CONCURRENT RAIL OPERATIONS**

#### **4.1 Overview**

This chapter describes the various rail operations within a port. The infrastructure as well as the various processes involved in the clearance of rail cargo is detailed in the first part of the chapter. The chapter discusses the simulation of concurrent rail operations within the architecture described in Chapter 3 and the methods used to resolve various issues involved. The chapter concludes with an analysis of the results of the simulation.

#### **4.2 Description of Rail Operations**

This section deals specifically with the process model used in CPORTS to deal with the clearance of rail cargo from a port. The first sub-section describes the infrastructure within the port that is used for rail operations. The following sub-sections describe the various processes involved in POD and POE rail operations.

##### **4.2.1 Operational Infrastructure**

Rail operations are concentrated within two areas in a port - the Interchange Yard Area (IYA) and the Rail Spur Area (RSA). An IYA is a holding area for empty trains waiting to be loaded with cargo as well as a reconstruction area for trains loaded with cargo. A RSA is usually smaller than an IYA and is further inside the port. It is a holding area for strings of rail cars being loaded with cargo.

#### 4.2.2 Port of Debarkation (POD) Operations

Rail cargo is brought into the port in the same manner as other types of cargo and is processed through the Berth Areas and the Staging Areas (SAs) in a similar fashion. At this stage, cargo is processed differently based on the mode of transport by which the cargo is scheduled to leave the port. Cargo scheduled to leave by highway transports is processed through a Loading Area and leaves through a gate. Cargo scheduled to leave by convoy is assembled in the Convoy Construction Area (CCA). The CCA can be inside or outside the port and cargo may have to pass through a gate to get to the CCA. Once a convoy has been assembled, it proceeds to its destination.

Rail cargo follows a different path through the port, which is shown in Figure 4.1 [12], [16]. Rail cargo is processed in the SA and is placed in a central pool for the entire set of staging areas called the rail cargo pool. The cargo waits in this rail cargo pool until a train arrives at the port to carry the cargo out of the port. When enough cargo to fill a train arrives in the SA, a train, if available is advanced to the IYA. At the IYA, the train is split into rail strings to fit the smaller RSAs for loading cargo. The cargo meant for a rail string is sent from the SA to the appropriate RSA. Once the rail string is loaded, it is pulled back to the IYA and the next rail string is sent to the RSA. This continues until the entire cargo is loaded onto the train. After it has finished loading, the train waits for a commercial locomotive to become available. The commercial locomotive then pulls the loaded train out of the port to its destination. Figure 4.2 shows the process-level interaction between various port areas during the flow of POD rail cargo through the port. The various POD processes and their interaction are described in detail in Appendix A.



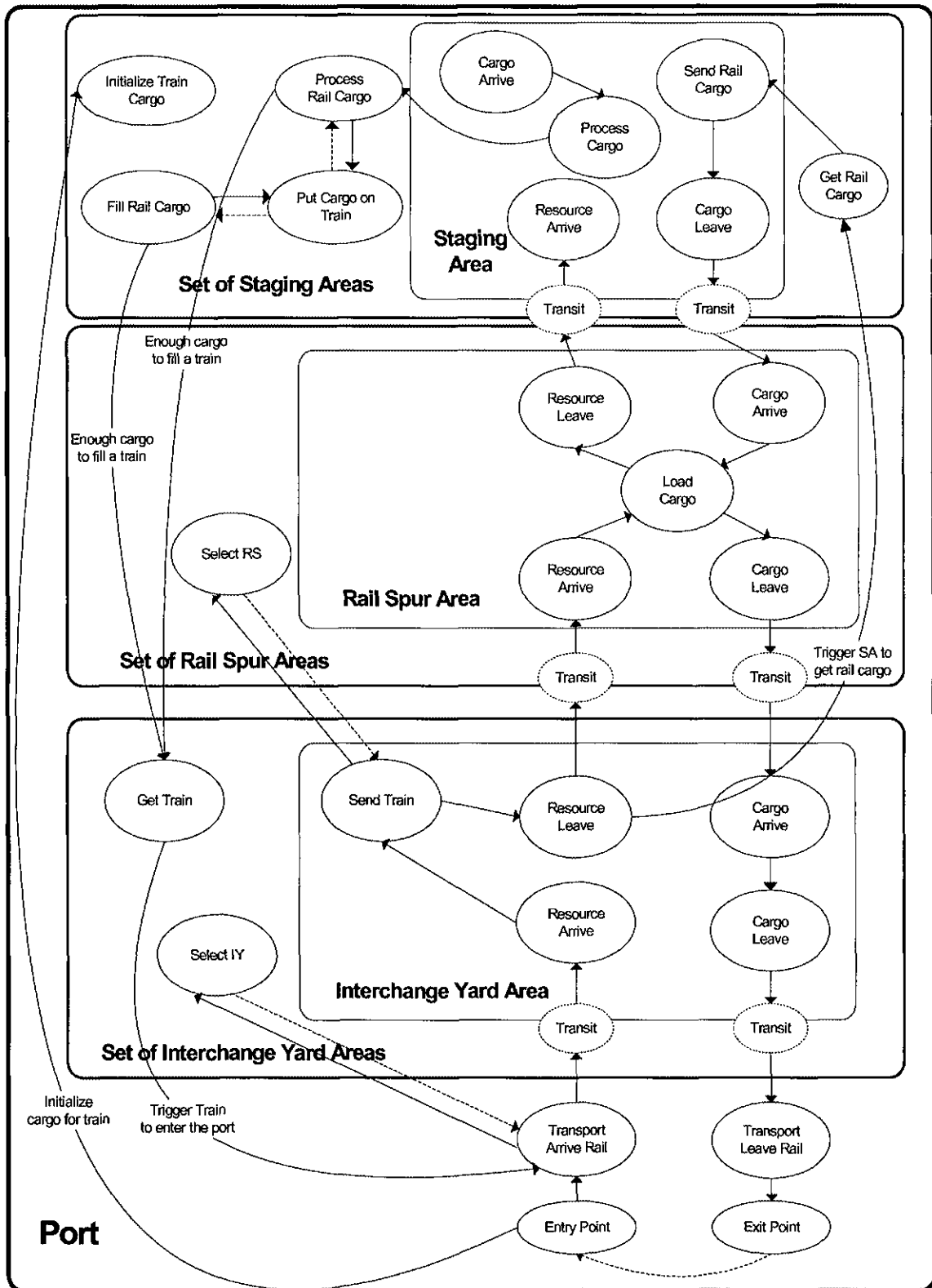


Figure 4.2 POD Rail Cargo Processes

### **4.2.3 Port of Embarkation (POE) Operations**

Rail cargo is brought into the port by trains that have been loaded with cargo at an installation or a POD port. These trains are then advanced to the next available IYA at the port. At an IYA, a train is split into rail strings to fit the smaller RSAs. At an RSA, the rail strings are unloaded using the resources available in that area. Resources arrive to transport the cargo from the RSA to a suitable SA. Once a rail string has been unloaded, it is pulled back to the IYA and the next rail string is sent to the RSA. This continues until the entire cargo has been unloaded from the train. After it has finished unloading, a train waits for a commercial locomotive to become available. The commercial locomotive then pulls the empty train out of the port to its destination. In the SA, rail cargo waits to be transported to the Berth Areas to be loaded onto ships, which are assigned to carry the cargo. Cargo is loaded onto the ships, which then clear the cargo from the port through the Anchorage Area.

This flow of POE rail cargo through a port is shown in Figure 4.3 [12], [16]. Figure 4.4 shows the process-level interaction between various port areas during the flow of POE rail cargo through a port. The various POE processes and their interaction are described in detail in Appendix B.

### **4.2.4 Resource Contention between POD and POE Operations**

The POE operations through a port directly affect the POD operations while both operations are being run concurrently. Specifically, the POE operations affect the

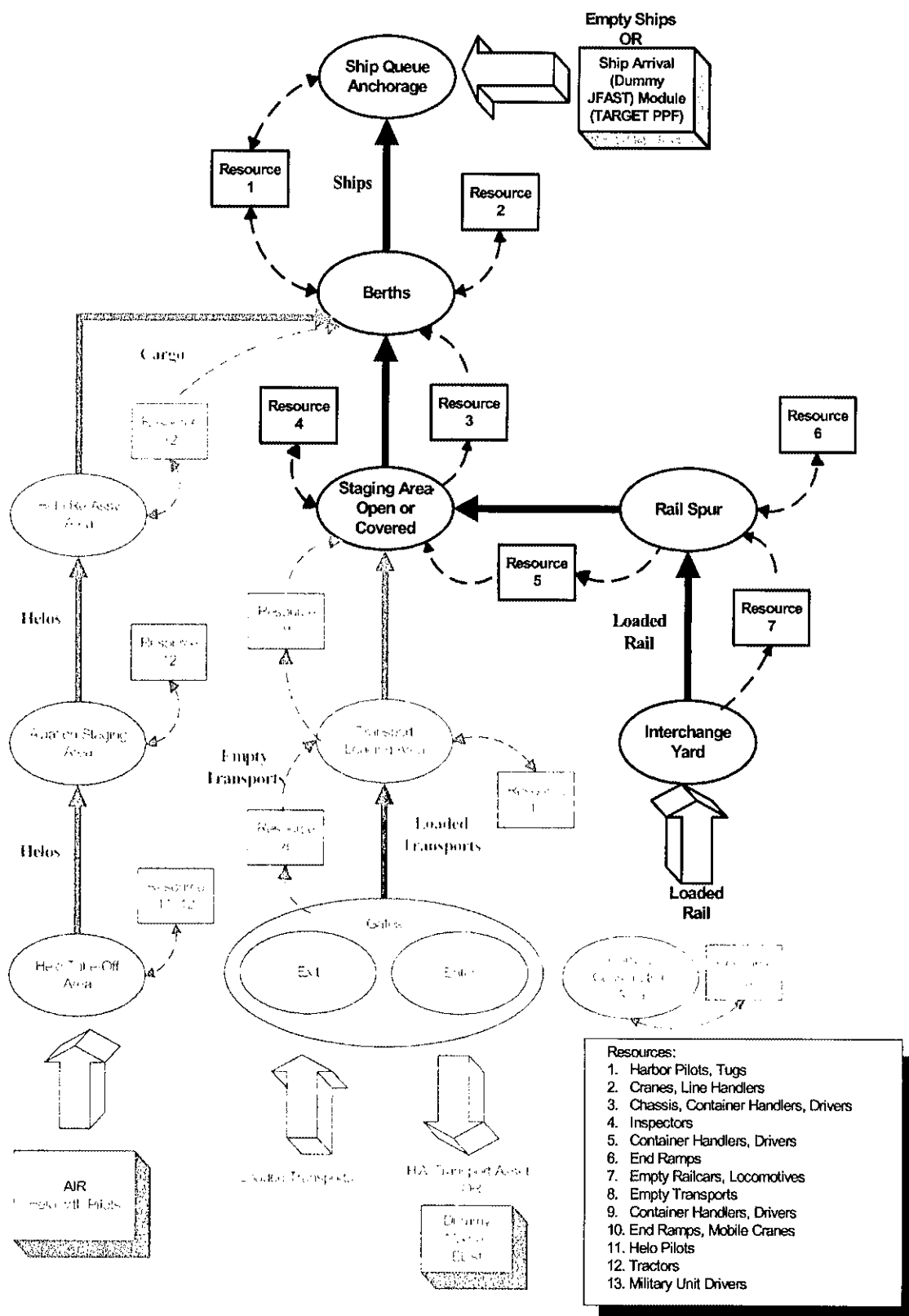
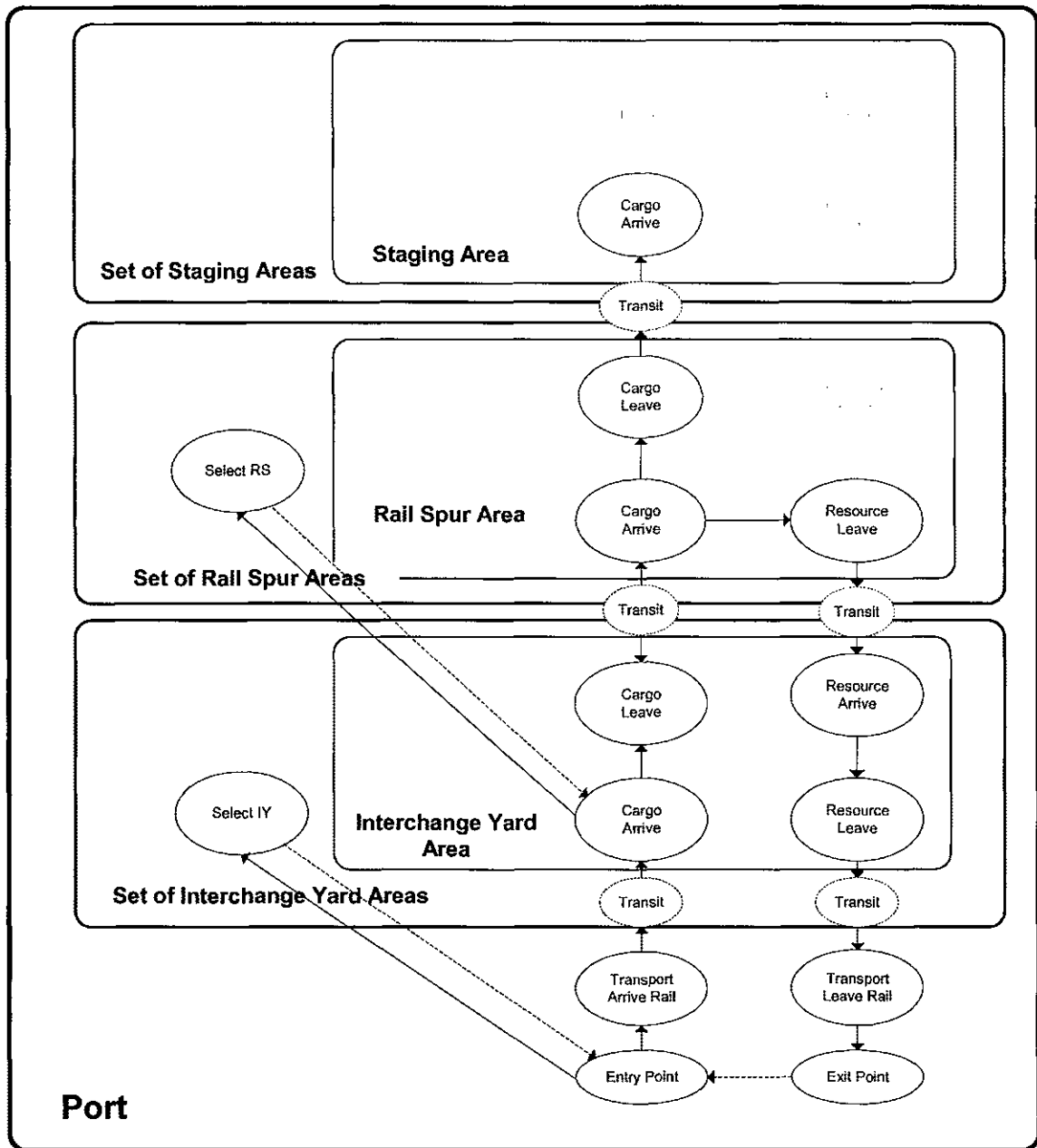


Figure 4.3 Flow of POE Rail Cargo through a Port



**Figure 4.4 POE Rail Cargo Processes**

clearance times for the POD operations due to the diversion of port infrastructure and resources for POE operations. The availability of the following infrastructure and resources are affected by the POE operations.

- **Port Infrastructure**

All infrastructure within the port has a finite storage area. POE rail operations increase the contention for the following port areas.

- *Interchange Yard Areas*

Trains carrying POE rail cargo and trains scheduled to carry POD rail cargo have equal priority to enter the IYA. The trains carrying POE rail cargo take up space in an IYA, delaying empty trains waiting to carry POD rail cargo out of the port.

- *Rail Spur Areas*

Rail strings carrying POE rail cargo and rail strings to be loaded with POD rail cargo have equal priority to enter the RSA. Since an RSA accepts only one rail string (POD or POE) at a time, rail strings waiting to be loaded with POD cargo are delayed by POE rail strings being unloaded, thereby affecting the clearance of POD rail cargo through the port.

- *Staging Areas*

All types of POD cargo (except helos and watercraft) wait in the SA for transports and resources to clear them out of the port. POE rail cargo that is brought into the RSA requests space in the SA. Since all requests for space (for POD or POE cargo) are granted on a first-come-first-served basis, POD cargo is on a ship for a longer time waiting for space to be freed up in the SA.

- *Berth Areas*

Ships loaded with POD cargo wait in the berth areas for the cargo to be unloaded. Ships arriving at the port to carry POE cargo out of the port also request space in the Berth Areas. Since all requests for space (for POD or POE ships) are granted on a first-come-first-served basis, POD ships may have to wait longer to dock at the port.

- **Port Resources**

Port resources are available on a limited basis. POE rail operations increase the contention for the following resources.

- *Port Locomotives*

Port locomotives haul rail strings between an IYA and an RSA. Locomotives required to haul POE rail strings reduce the number of locomotives available for rail strings carrying POD cargo.

- *Staging Area Resources*

Container handlers and drivers are required to transport cargo between a SA and an RSA. POE rail cargo increases the requests for these resources, thereby delaying POD cargo waiting in the SA to be transported to the RSA.

- *Berth Area Resources*

Harbor pilots, tugs, cranes, line handlers, chassis, container handlers, and drivers are required to transport cargo between a SA and a Berth Area. POE cargo increases the requests for these resources, thereby delaying POD cargo waiting in the Berth Area to be transported to the SA.

A potential deadlock situation that might arise is the scenario that the SA is filled with POD rail cargo waiting for trains to carry it out of the port, while the IYAs and RSAs in the port are filled with POE rail strings and trains waiting for space to be freed up in the SA so that cargo can be unloaded from the rail strings and the POE trains can leave the port. In this case, the simulation is deadlocked with both POD and POE operations being stalled for need of infrastructure and / or resources. The deadlock can be resolved operationally by increasing the available area in the port areas, particularly the SA, by adjusting the arrival profile of cargo and / or transports entering the port or by partitioning infrastructure and resources based on the mode of cargo (POD or POE).

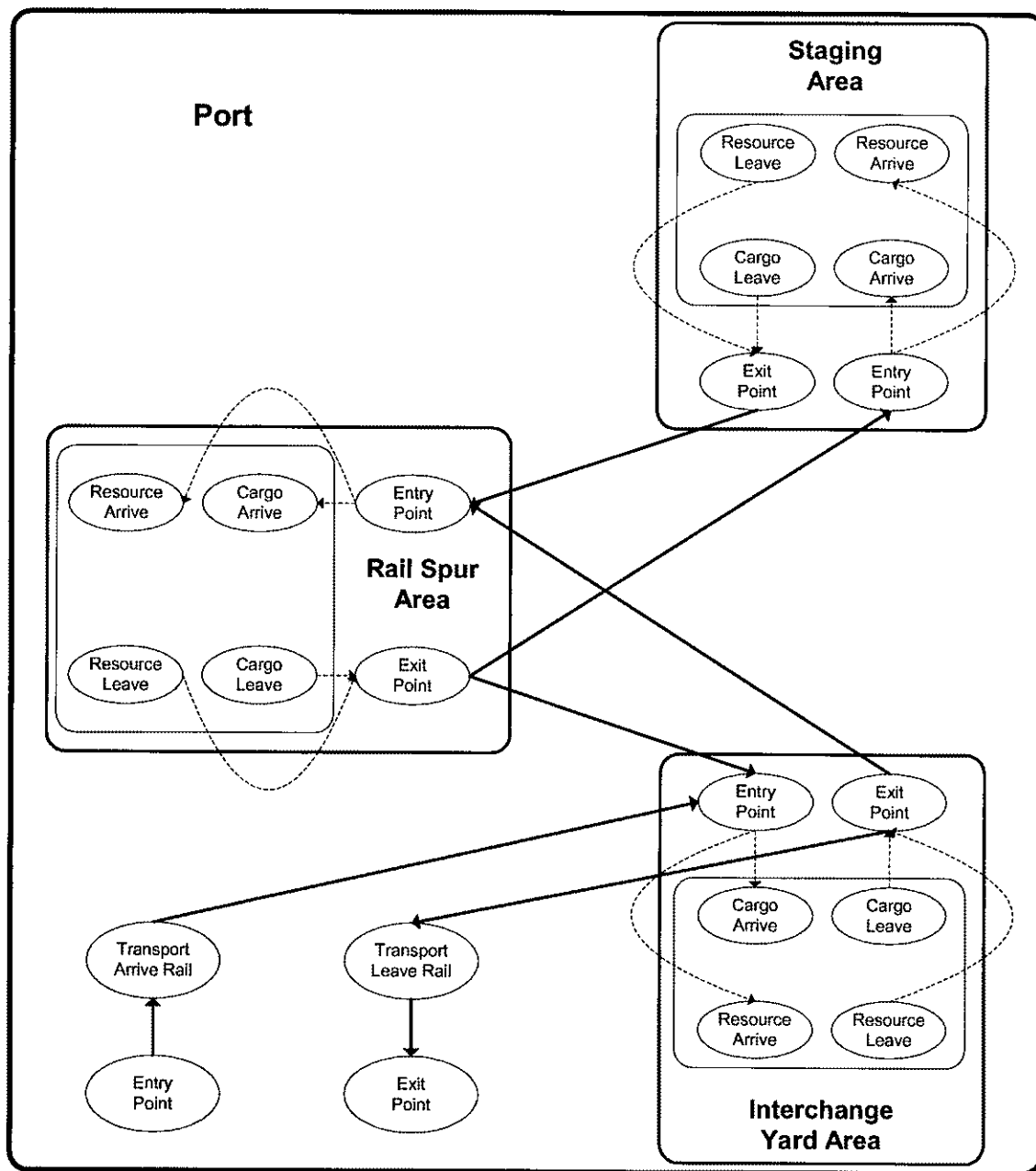
### **4.3 Simulation of Concurrent Rail Operations**

This section deals with the simulation of concurrent rail operations through a port. It shows the ease with which the POE rail operations can be simulated concurrently with the POD rail operations using the architecture described in Chapter 3. The first part of this section deals with the interconnection between components within the model. The second part of the section discusses how resource and infrastructure contention is resolved in the model.

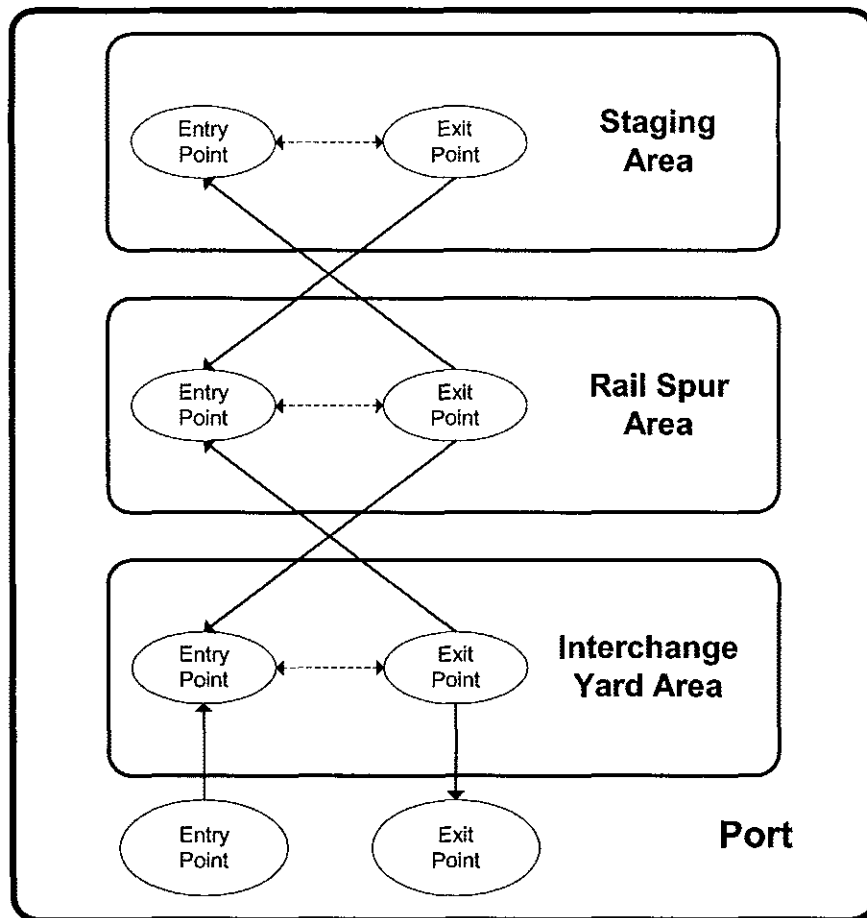
#### **4.3.1 Concurrent Rail Operations**

POD operations are first modeled using the architecture described in Chapter 3. POE rail operations are then integrated into the same architecture to model the concurrent flow of rail cargo through the port. POE rail operations are modeled from the arrival of trains at the port loaded with cargo till the SA, for demonstration purposes. A complete

concurrent POD / POE simulation can be developed by extending the POE process model to model the flow of cargo until they are loaded on to ships and by including other types of cargo arriving by different transport modes.



**Figure 4.5 Micro-Level Interconnection between Port Areas for Concurrent POD / POE Rail Operations**



**Figure 4.6 Macro-Level Interconnection for Concurrent POD / POE Rail Operations**

Figures 4.2 and 4.4 show the processes involved in the flow of rail cargo through a port. Both POD and POE resources and transports follow complementary paths through the port. The interconnection between port areas is common for both the POD and POE rail operations. Figure 4.5 shows the micro-level interconnection between various port areas during the flow of cargo. At a higher level of abstraction, resources and transports enter and leave a port area through the *EntryPoint* and *ExitPoint* respectively. A resource or a transport enters a port area and is processed based on the *process mode* of the cargo (POD or POE) it is carrying or is scheduled to carry at some point in the future. The

architecture is, therefore, flexible and robust enough to integrate both the POD and POE rail processes. Figure 4.6 shows the macro-level interconnection for both processes.

#### **4.3.2 Resolving Resource and Infrastructure Contention**

An important aspect of integrating POD and POE operations into the same architecture is to resolve the resource and infrastructure contention within the port, which might not appear for a POD operation. It is unavoidable that the POE operations delay the clearance of POD cargo through the port. However, it is vital that the POE operations do not stall all operations within the port. The contention between various resources is resolved in the following manner.

- **Port Infrastructure**

- *Interchange Yard Areas*

Trains carrying POE rail cargo and trains destined to carry POD rail cargo, both compete for common space in the IYAs. Once an IYA has been selected, space is requested in that IYA. The allocation of space indicates that space is available and that the space has been reserved for the train. When the train leaves the IYA, it releases its space.

- *Rail Spur Areas*

Rail strings carrying POE rail cargo and rail strings to be loaded with POD rail cargo both compete for common space in the RSAs. Since an RSA accepts only one rail string (POD or POE) at a time, once an RSA has been allocated, no other requests

for space in the RSA are granted. When the rail string leaves the RSA, it releases its space indicating that space is available for the next rail string.

- *Staging Areas*

Port areas in general have limited capacities. Multiple pieces of cargo can be concurrently transiting to a SA, each requiring space. The cargo arriving at the SA first occupies the space, with the other(s) being forced to wait outside the SA. This could result in an infinite queue. To address this, port areas have a request space capability. Before the cargo begins to transit to a port area, space is requested in the port area. The granting of space not only indicates that space is available, but that space has been reserved for the cargo. When the cargo leaves the port area, it releases its space.

- **Port Resources**

- *Port Locomotives*

Locomotives are kept in a single resource pool within the port. Locomotives are requested from the same central pool to haul both POD and POE rail strings between RSAs and IYAs and then are released back into the same pool.

- *Staging Area Resources*

Container handlers and drivers required to transport cargo between a SA and an RSA are kept in a single resource pool (one for each) within the port. Both POD and POE rail cargo request resources from the same pools.

#### 4.4 Analysis and Results

This architecture for concurrent POD / POE processes is tested by simulating the flow of concurrent rail cargo through the port of Ad Dammam in Saudi Arabia. The Ad Dammam scenario has been built to simulate the POD operations through the port for the CPORTS project. As mentioned before, all the resources and infrastructure of the port generally are not available for military operations. A specified portion of the port is allocated for military operations. Certain modifications need to be made to the scenario usually used for POD operations to accommodate the simulation of POE rail operations concurrently with the POD operations:

- A separate executable file is built to simulate the concurrent flow of rail cargo.
- The POD cargo consists of only rail cargo and the same is true for the POE cargo.
- The SA capacity in the port of Ad Dammam that is available for POD operations has been increased by a factor of 1.3 to handle the extra load of the POE rail cargo as well to avoid any possible deadlock condition that might occur within the simulation.
- Resources and infrastructure within the port are allocated on a first-come-first-served basis for both POD and POE operations.

Figure 4.7 shows the routes between various port areas as well as the flow of different types of POD cargo through the port of Ad Dammam [13], [16]. Figure 4.8 shows the POE cargo flow. The simulation of concurrent operations involves flow of POD and POE cargo simultaneously.

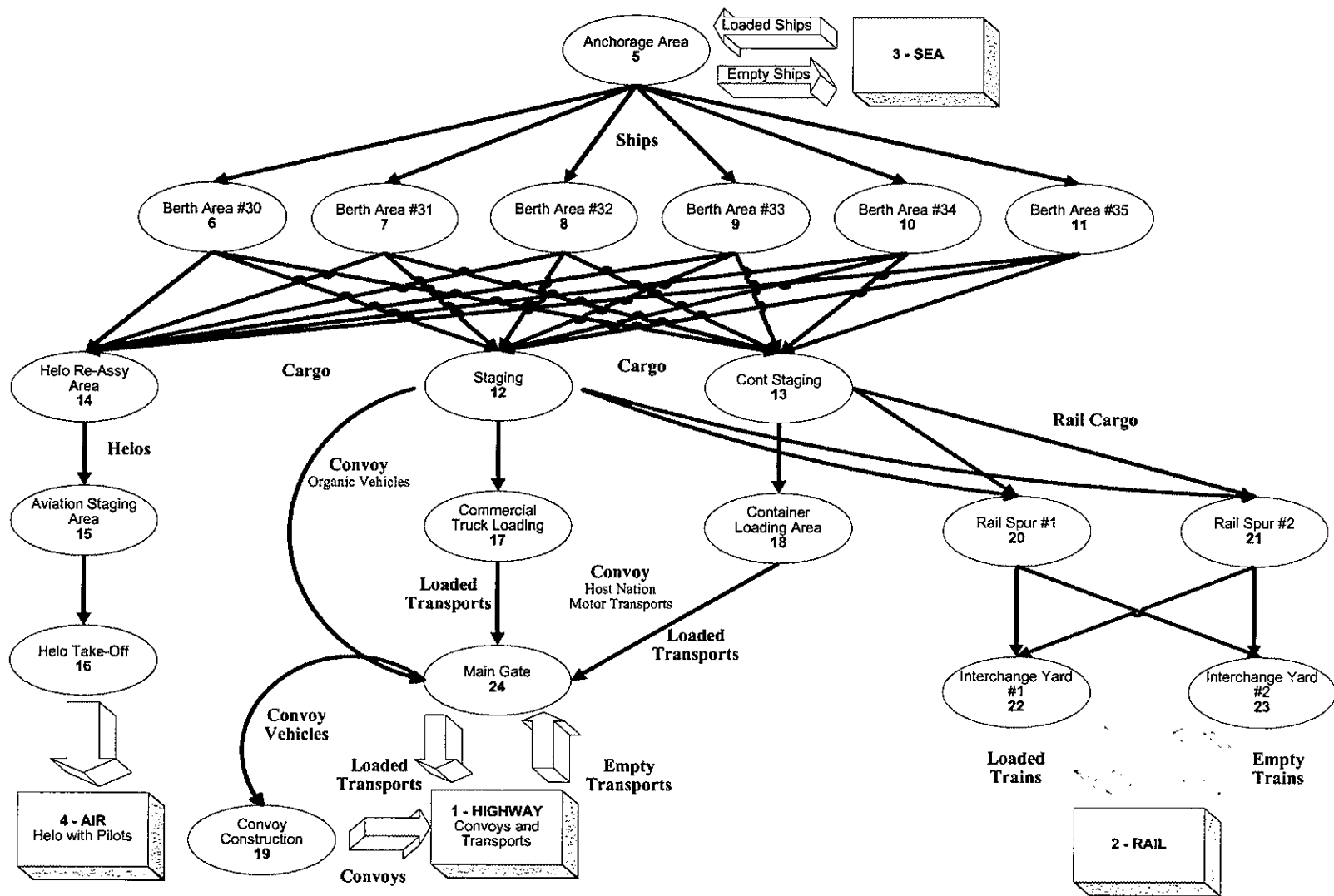


Figure 4.7 CPORTS POD Scenario File Routes for Port Ad Dammam

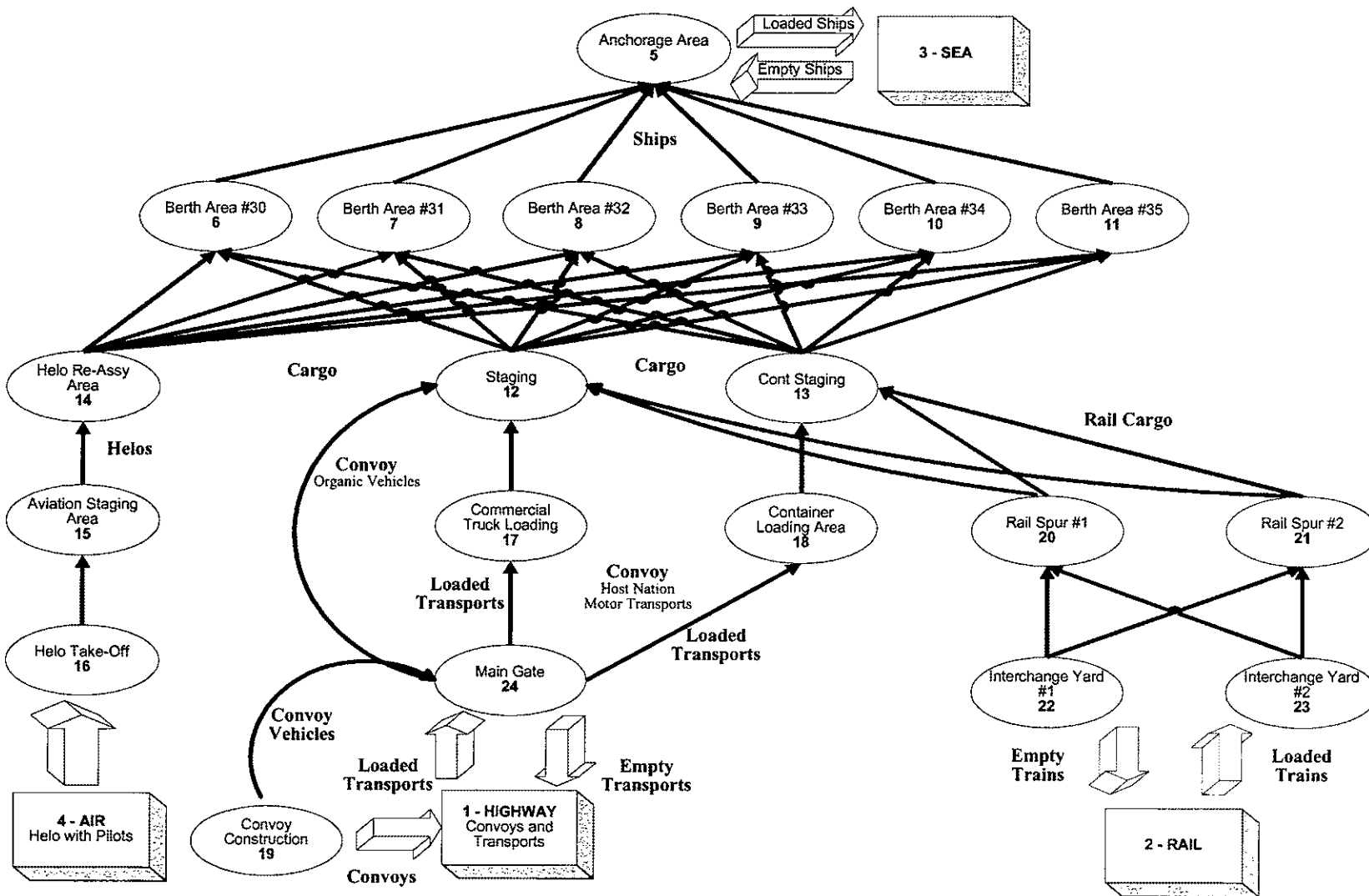
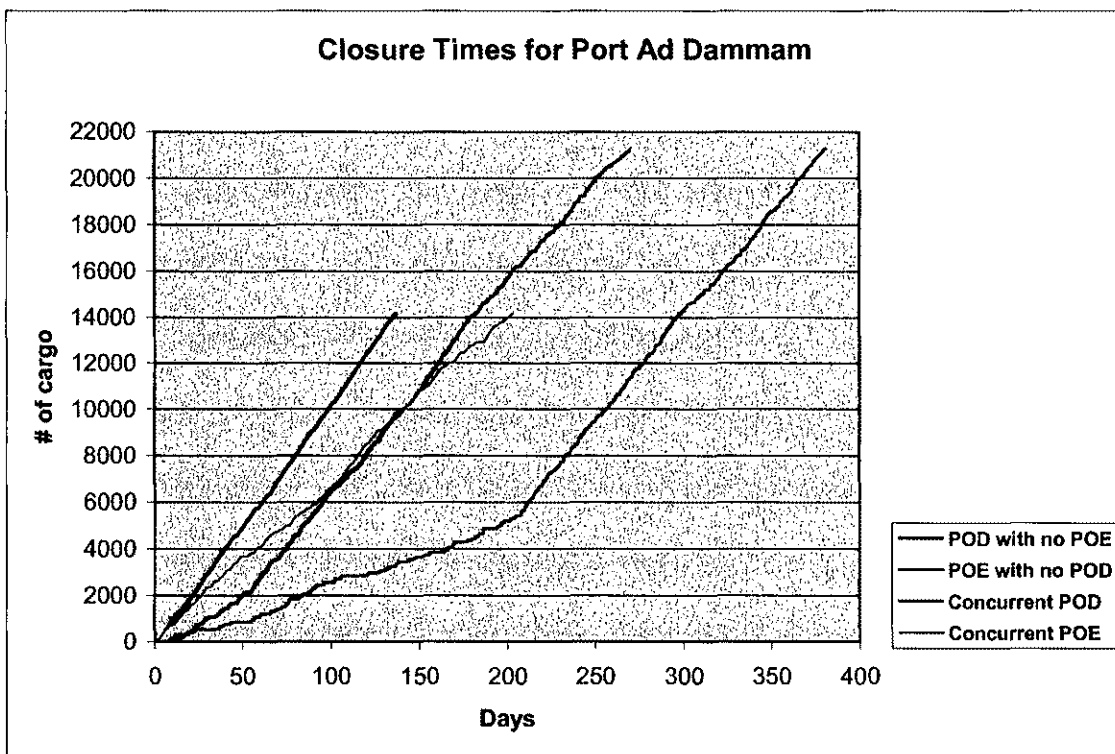


Figure 4.8 CPORTS POE Scenario File Routes for Port Ad Dammam

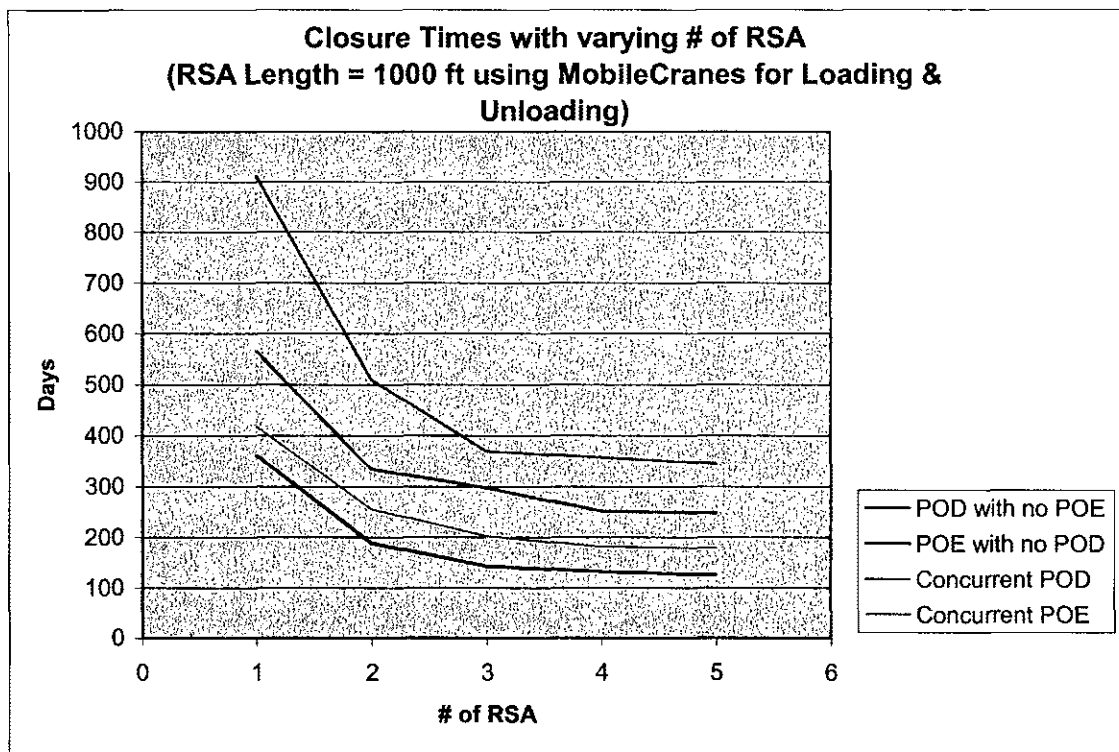
The details of the simulation are given below:

- Number of POD rail cargo = **21264 pieces**.
- Empty trains to carry POD cargo out of the port arrive with an interarrival time that has an *exponential distribution* with a mean of **100 hours**.
- Empty POD trains return to the port to carry more cargo after a cycle time that has an *exponential distribution* with a mean of **350 hours**.
- Number of POE rail cargo = **14150 pieces**.
- Trains carrying POE cargo arrive once every **24 hours** (constant).
- Both POD and POE trains have *equal priority* to enter the port and to be allotted space in the IYAs and RSAs.



**Graph 4.1 Closure Times for Port Ad Dammam for various Cargo flows**

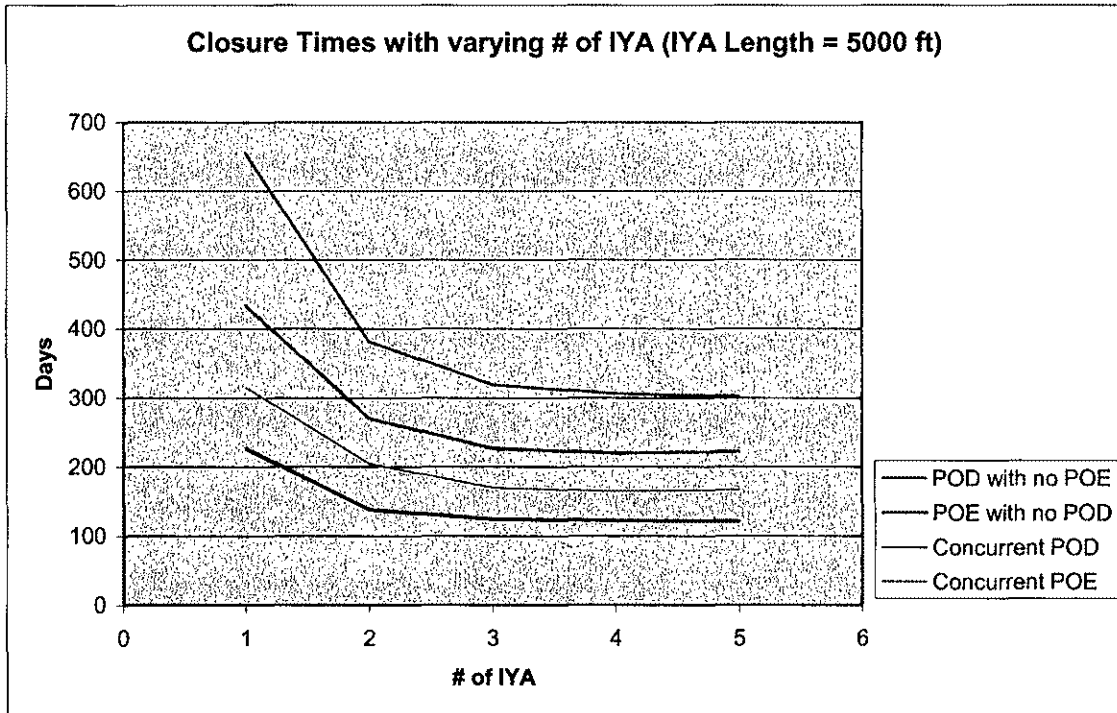
Graph 4.1 shows the closure times for the POD cargo and POE cargo by themselves as well as the closure times for the concurrent flow of POD and POE cargo. The graph shows that clearance of POD cargo is affected by the POE cargo. After all the POE cargo has been cleared through the port, the POD cargo continues to be cleared at the same rate as when only POD cargo flows through the port.



**Graph 4.2 Closure Times for various Cargo flows with varying number of Rail Spur Areas**

Graph 4.2 shows the effect of the number of RSAs on the clearance of cargo through the port. The graph shows that as the number of RSAs increases, the clearance times for all flows of cargo decreases. The graph also shows that four RSAs are enough to clear this particular cargo scenario.

Graph 4.3 shows the effect of the number of IYAs on the clearance of cargo through the port. Similar to the previous graph, it shows that as the number of IYAs increases, the clearance times for all flows of cargo decreases. Since only rail cargo is involved in this particular simulation run, the number of RSAs and IYAs has the most impact on the clearance of cargo.



**Graph 4.3 Closure Times for various Cargo flows with varying number of Interchange Yard Areas**

The performance of the simulation in terms of execution time depends on the configuration of the machine being used. Table 4.1 shows the execution times for the POD simulation and the POE simulation, as well as the simulation of the concurrent flow of cargo on different machines. The table shows that the concurrent simulation does not add significant overhead to the POD simulation above the required POE simulation time.

No.	Configuration (Processor Speed / RAM size)	Execution Time for POD Simulation (seconds)	Execution Time for POE Simulation (seconds)	Execution Time for Concurrent Simulation (seconds)
1	Pentium 4 (2.4 GHz / 1 GB)	7	2	9
2	Pentium 4 (2 GHz / 1 GB)	9	2	11
3	Pentium 4 (1.4 GHz / 512 MB)	14	3	16
4	Pentium 3 (1 GHz / 512 MB)	12	4	16
5	Pentium 3 (667 MHz / 256 MB)	18	5	23
6	Pentium 3 (500 MHz / 256 MB)	24	6	30

**Table 4.1 Execution Times for the Simulation**

The results of the analysis can be summarized as follows:

- POE operations have a significant impact on POD operations. This impact is important when the flow of POD cargo is mainly for sustainment of existing military forces, while the POE cargo is mainly cargo returning after deployment.
- The architecture is robust and flexible enough to handle concurrent operations.
- The simulation of concurrent operations for a relatively large scenario executes within 30 seconds on any standard computer.
- The model can support the simulation of 500,000 pieces of cargo, which adequately satisfies the stated requirement of the model.

## *Chapter V*

### **CONCLUSION**

Transportation logistics simulation is extensively used by the military to plan for the transportation of military cargo, i.e., troops, equipment, and supplies, to the Theatre of War. The architecture described in this thesis provides a means of analyzing the concurrent flow of cargo in opposite directions, both to and from the Theatre of War, through a cargo terminal and eventually through a network of cargo terminals. Although this architecture has been developed for military applications, it can also be used for commercial applications to study the flow of commercial cargo through a network of cargo terminals to its final destination. This chapter details the achievements of this thesis and suggests several enhancements.

#### **5.1 Achievements**

The work described in this thesis document has achieved the following:

- It is the first architecture that supports the simulation of concurrent POD and POE military operations.
- The architecture itself is object-oriented and hence exhibits all the advantages of object-oriented programming.
- The simulation of concurrent operations shows the effect that POE processes have on POD processes. The clearance times, as well as the availability of resources and infrastructure for POD operations, are clearly affected by the POE operations.

- The simulation of concurrent operations enables the port expert to devise methods to support concurrent operations without adversely affecting either of the operations.
- The time required for the simulation of concurrent operations is on average no worse than the sum of the simulation times for individual POD and POE operations.
- The architecture can support the simulation of 500,000 pieces of cargo flowing in both directions.
- This architecture is currently being used for the simulation of a network of cargo terminals and has proved to be extremely flexible for development purposes.

## **5.2 Enhancements**

The architecture described in this thesis provides a framework to support the simulation of concurrent POD / POE operations through a port. The architecture is shown to be robust enough to support concurrent rail operations through a port. It is hoped that the architecture would support a full-scale concurrent operation through a port.

Towards that end, a few possible enhancements are listed below.

- The architecture described in this document has been tested with a full POD flow of rail cargo and a minimal, restricted flow of POE rail cargo. A full test can be performed by describing the full POE processes until cargo is loaded onto ships, which then leave the port.

- The architecture can be tested further by incorporating other transport modes like Highway, Air and Convoys into the POE process model.
- The architecture can be used to initialize multiple cargo terminals at the same time and to connect them together to form a network of terminals. This will help to study an end-to-end cargo flow in both directions.
- This architecture has been developed in the MODSIM simulation language [14], [15]. Due to the limited lifetime of the MODSIM language, there is a need to implement the architecture in other object-oriented languages like C++ or Java to support future projects.

Overall, the architecture has been successful in meeting all the requirements for the simulation of concurrent POD / POE operations through a port. The success of the architecture has been borne out further by the fact that the Military Traffic Management Command Transportation Engineering Agency (MTMCTEA) is funding the effort for a complete POD / POE model to be built explicitly to adhere to this architecture.

## REFERENCES

- [1] Leathrum, J., T. Frith, R. Mathew et al. 2002. "An Object Architecture for the Simulation of Networks of Cargo Terminal Operations," *In Preparation*.
- [2] Leathrum, J. and T. Frith. 2000. "A Reconfigurable Object Model for Port Operations," *Proceedings of the Summer Computer Simulation Conference, SCSC 2000*: 603~608.
- [3] Hayuth, Y., Pollatschek, M.A., Roll, Y. 1994. "Building a Port Simulator," *Simulation* 70, no. 3 (Mar.): 179~189.
- [4] Merkuryew Y., Tolujew, J., Blumel, E. 1998. "A Modeling and Simulation Methodology for Managing the Riga Harbour Container Terminal," *Simulation* 71, no. 2 (Feb.): 84~95.
- [5] Gambardella, L.M., Rizzoli, A.E., Zaffalon, M. 1998. "Simulation and Planning of an Intermodal Container Terminal," *Simulation* 71, no. 2 (Feb.): 107~116.
- [6] Ramani, K.V. 1996. "An Interactive Simulation Model for the Logistics Planning of Container Operations in Seaports," *Simulation* 66, no. 5 (May): 291~300.
- [7] Bruzzone, A., Signorile, R. 1998. "Simulation and Genetic Algorithms for Ship Planning and Shipyard Layout," *Simulation* 71, no. 2 (Feb.): 74~83.
- [8] Nevins, M., C. Macal and J. Joines. 1998. "A Discrete-Event Simulation Model for Seaport Operations," *Simulation* 70, no. 4 (Apr.): 213~223.
- [9] Nevins, M., C. Macal and J. Joines. 1995. "PORTSIM: An Object-Oriented Port Simulation," *Proceedings of the Summer Computer Simulation Conference, SCSC 1995*: 160~165.

- [10] Braun, M., G. Lurie, K. Simunich et al. 2000. "ELIST8: A Simulation System for Transportation Logistics Planning Support," *Proceedings of the Summer Computer Simulation Conference*, SCSC 2000: 693~698.
- [11] *Port Operational Performance Simulator (POPS)* User Manual, Version 1, 1987.
- [12] T. Frith, Leathrum, J., R. Mathew et al. 2002. "CPORTS Process Model," [http://www.vmasc.odu.edu/portsim/CPorts\\_Process\\_Model.pdf](http://www.vmasc.odu.edu/portsim/CPorts_Process_Model.pdf)
- [13] T. Frith. 2000. "CPORTS Port Scenario Layout."
- [14] *MODSIM-III Reference Manual*, CACI Products Company, La Jolla, CA.
- [15] *MODSIM-III Tutorial*, CACI Products Company, La Jolla, CA.
- [16] *Microsoft Visio 2000*, Microsoft Corp., Redmond, WA.

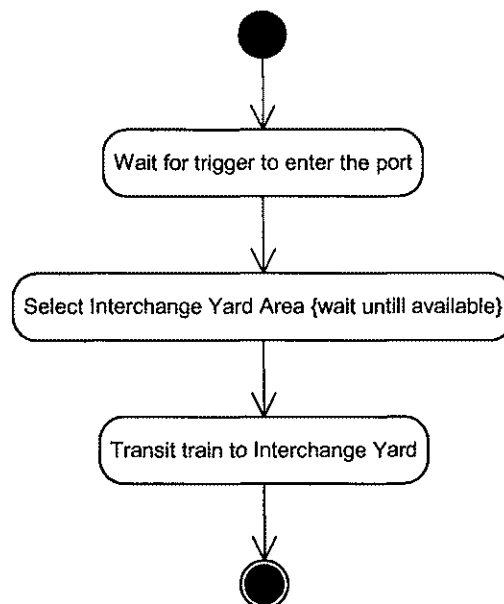
## APPENDICES

### A. POD Rail Processes

The processes associated with the clearance of POD rail cargo through a port can be described in two ways. The first is by studying the scenario of an empty train arriving at the port, getting loaded with cargo waiting to be loaded and leaving the port filled with cargo. The second is by studying the scenario where the cargo arrives before a train is available to carry it out of the port or there are trains waiting for cargo to fill them so that they can leave the port. Both are described in detail below.

#### A.1 Selecting cargo to fill a single train

The process of handling a train arriving at the port is shown in Figure A.1.



**Figure A.1 Model Flow Diagram for POD *TransportArriveRail* process in a *Port* Object**

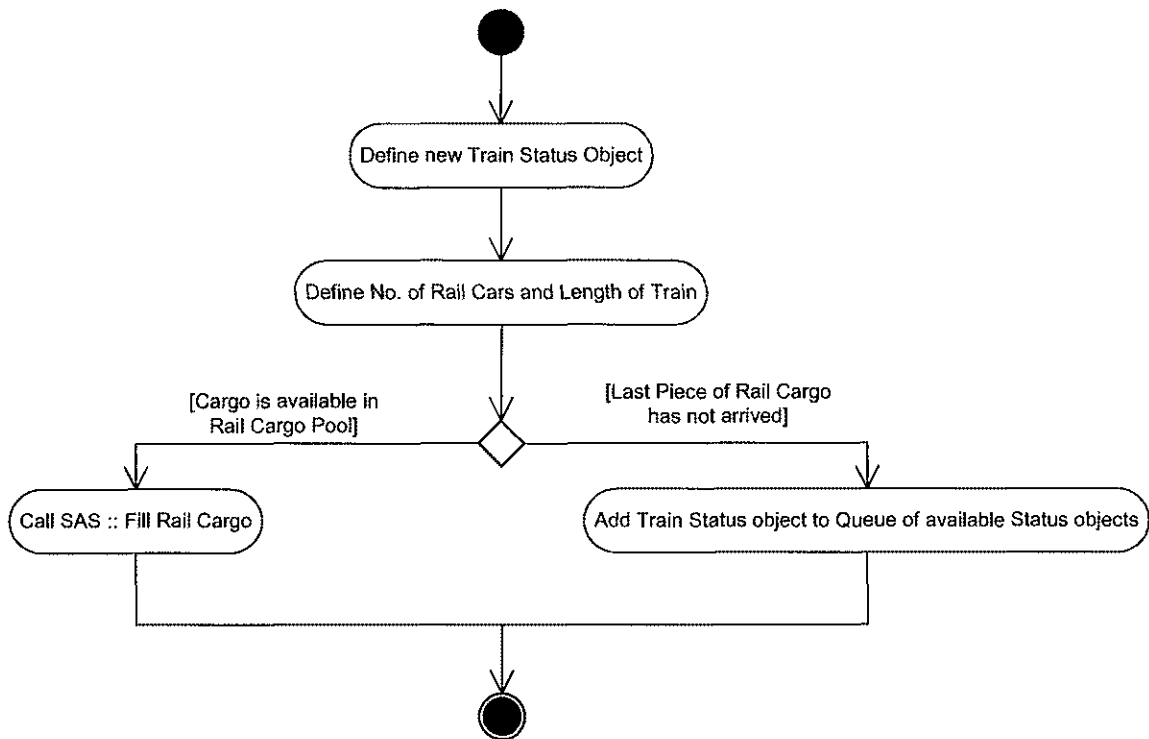
When the train arrives at the port, it does not directly enter the port. It calls the *InitTrainCargo* process in the *StagingAreaSet* object (Refer to Figure 4.2), which in turn triggers the rail cargo pool in the set of Staging Areas (SAs) to define enough cargo, which will fit in that train. In the *InitTrainCargo* process, a new *TrainStatus* object is defined for the train. The *TrainStatus* object is a clone of the train that has just arrived at the port and is used to allot cargo to the train without having the train enter the port. The *TrainStatus* object is initialized with all the characteristics of the train. Figure A.2 shows a *TrainStatus* object.

POD Port Process Model::Train Status
-Current Rail Car : Integer
-Current Rail Car Length : Integer
-Current Rail Car Capacity : Integer
-Status Init(in train)
-Init Current Car Length(in train)
-Init Current Car Capacity(in train)
-Calculate Current Car Length(in cargo)
-Calculate Current Car Capacity(in cargo)
-Decrement Car()

**Figure A.2 *TrainStatus* Object**

Cargo, if available is slotted to be loaded onto the train by calling the *FillRailCargo* process in the *StagingAreaSet* object (Refer to Figure 4.2). If no rail cargo is available to be loaded, the train (i.e., the respective *TrainStatus* object) is added to a queue waiting for rail cargo. The *InitTrainCargo* process is shown in Figure A.3.

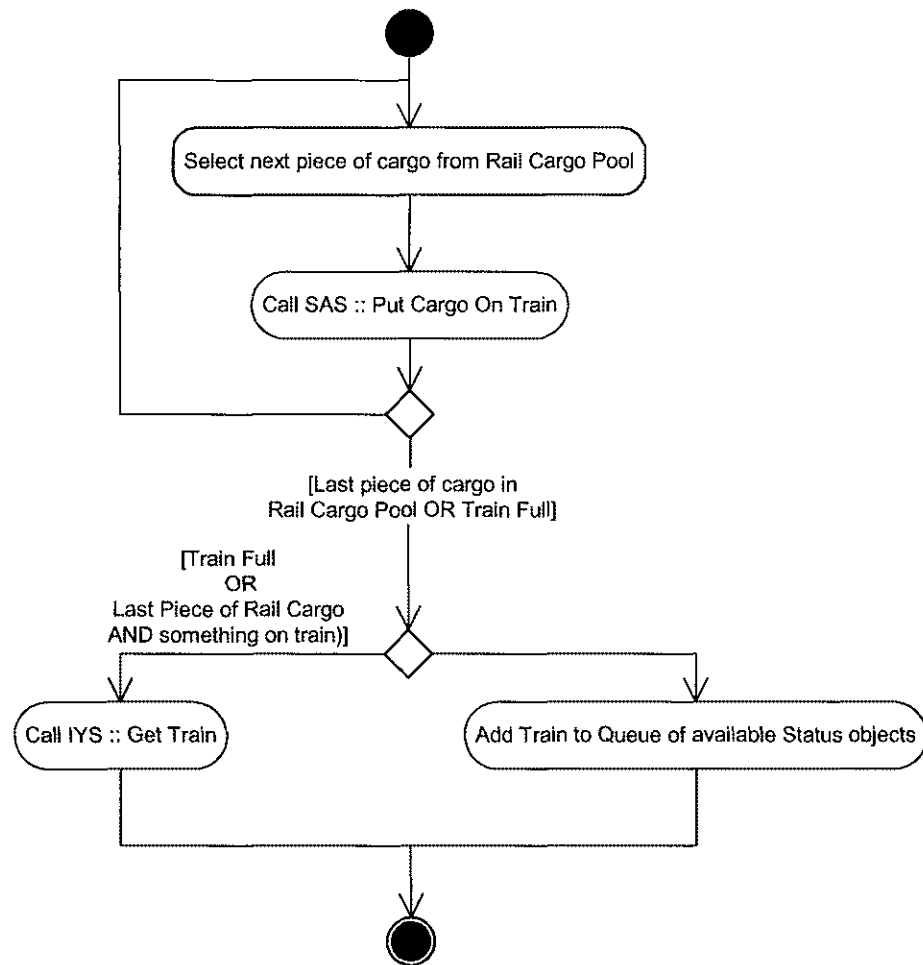
In the *FillRailCargo* process, cargo that is available to be loaded on the train is pulled out from the rail cargo pool and is passed on to the *PutCargoOnTrain* process in the *StagingAreaSet* object (Refer to Figure 4.2) to see if that particular piece of cargo can



**Figure A.3 Model Flow Diagram for POD *InitTrainCargo* process in a *StagingAreaSet* Object**

fit on the train. If the piece of cargo fits on the train, it is scheduled to be put on the train by adding it to the *TrainStatus* object. If the train is not filled up completely and no further rail cargo is currently available to be loaded, the train (i.e., its *TrainStatus* object) is added to a queue where it waits for further cargo. The *FillRailCargo* process is shown in Figure A.4.

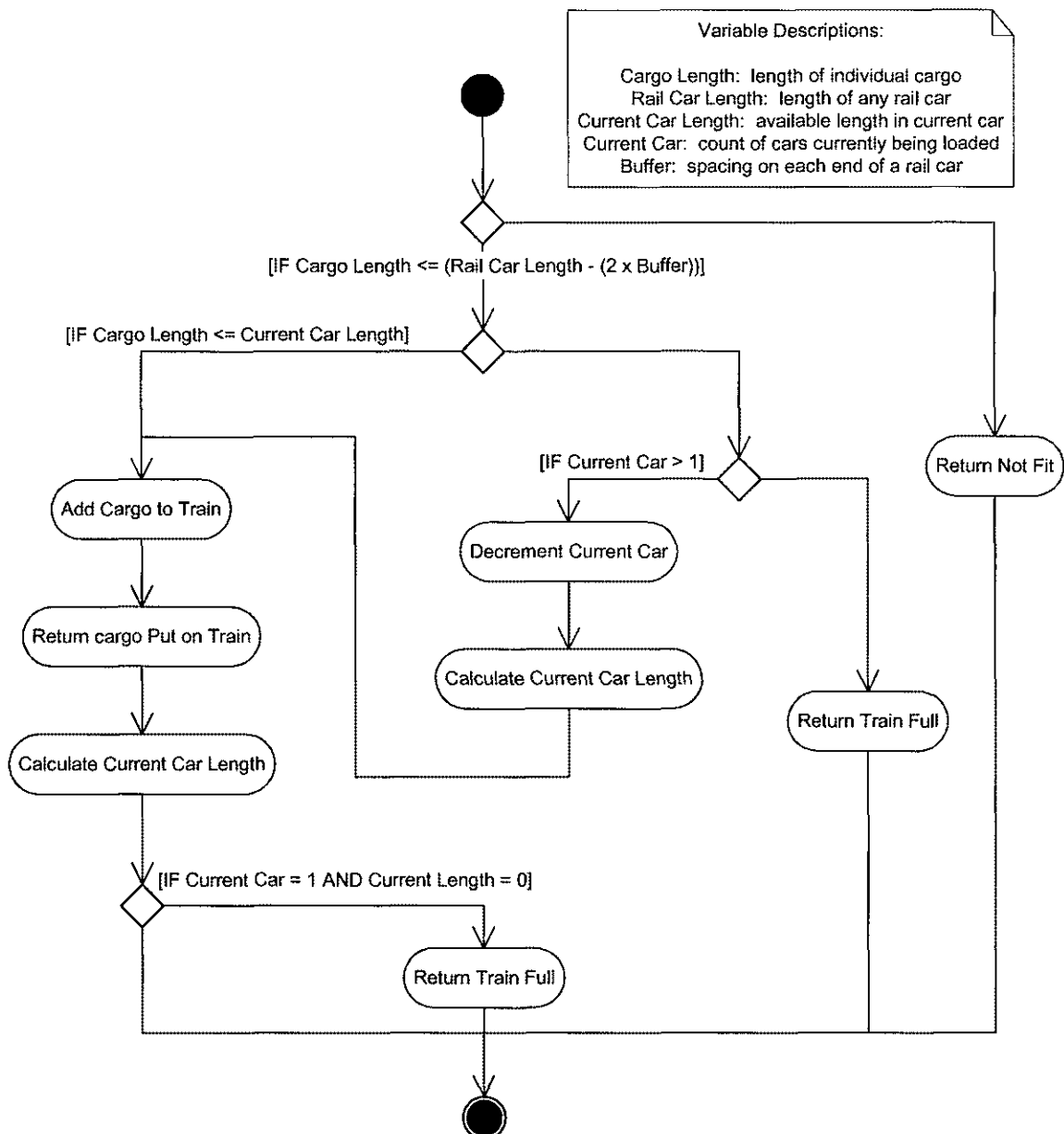
In the *PutCargoOnTrain* process, it is verified whether the piece of cargo will fit on the train. Appropriate allowance is made for buffering at the ends of a rail car as well as a buffer between two pieces of cargo. If the piece of cargo does not fit on the train and no other train is available to be loaded, the piece of cargo is added back to the rail cargo



**Figure A.4 Model Flow Diagram for POD *FillRailCargo* process in a *StagingAreaSet* Object**

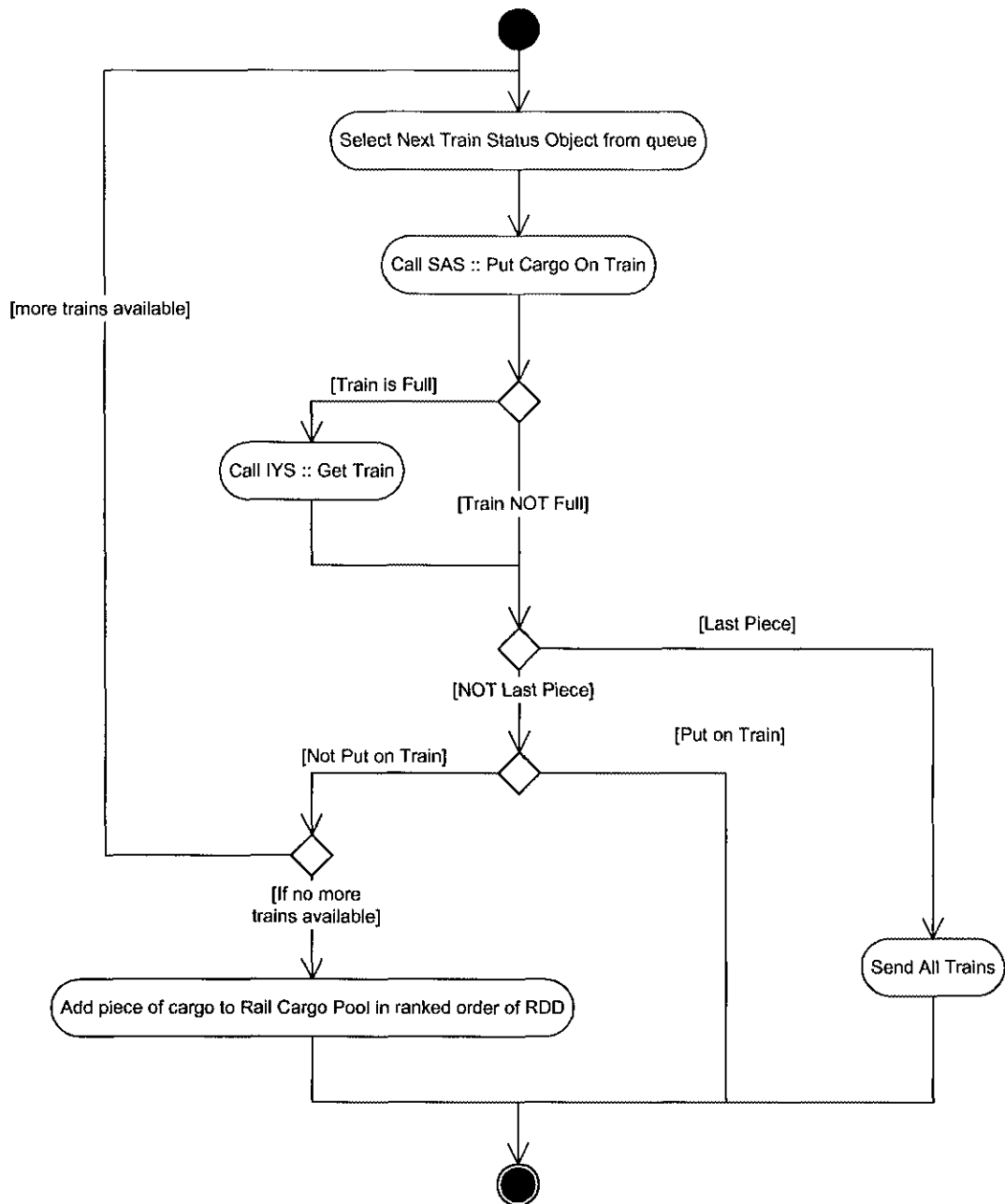
pool where it waits for another train to be loaded. The *PutCargoOnTrain* process is shown in Figure A.5.

The second way of describing the processes associated with the clearance of POD rail cargo through a port is by studying the scenario where the cargo arrives before a train is available to carry it out of the port or there are trains waiting for cargo to fill them so that they can leave the port. In both cases, the SA calls the *ProcessRailCargo* process in the *StagingAreaSet* object (Refer to Figure 4.2). This process is shown in



**Figure A.5 Model Flow Diagram for POD *PutCargoOnTrain* process in a *StagingAreaSet* Object**

Figure A.6. The piece of rail cargo that just arrived in the SA is checked to ascertain whether it can fit on any of the waiting trains (if any available), which already have cargo available to be loaded. The trains waiting for cargo are represented by their respective *TrainStatus* objects. If adding the piece of cargo fills up a train, then the train is triggered to enter the port using the *GetTrain* process in the *InterchangeYardAreaSet*

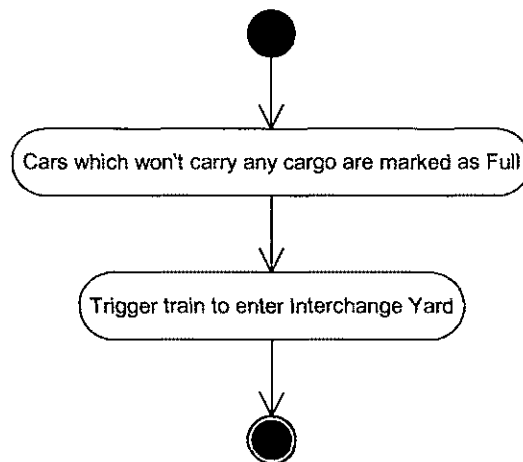


**Figure A.6 Model Flow Diagram for POD *ProcessRailCargo* process in a *StagingAreaSet* Object**

object (Refer to Figure 4.2). This process is explained in the next section. If the piece of cargo cannot be fitted on any available train, it is added to the rail cargo pool waiting for a train that can carry it out of the port.

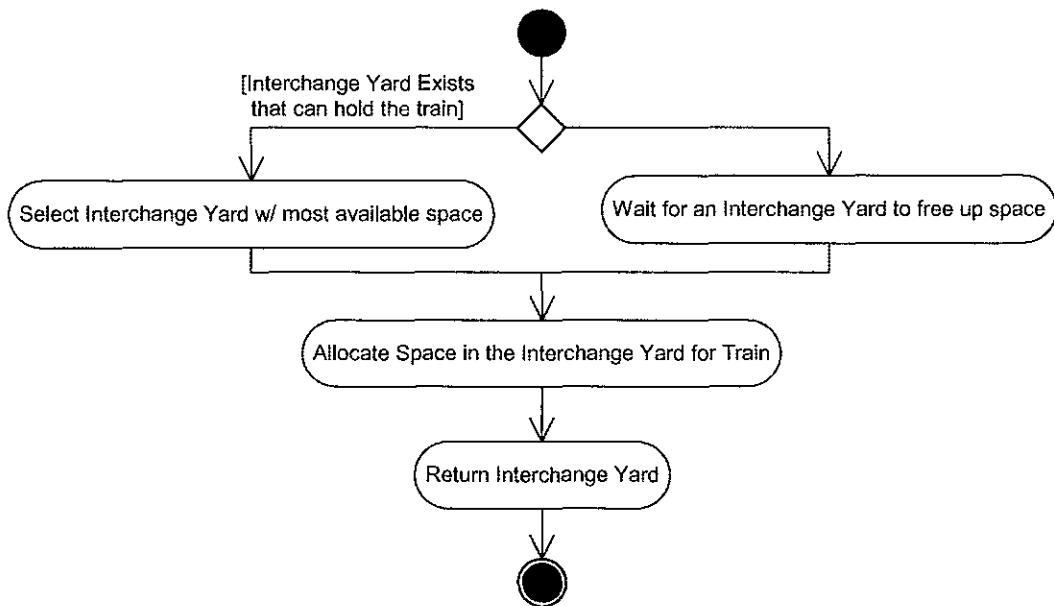
## A.2 Triggering a train to enter the port

Once enough cargo has arrived in the SAs to fill a train or if the last piece of rail cargo has arrived, the train is triggered to enter the port. This is done using the *GetTrain* process. The cars that are not scheduled to carry any cargo are marked as full before the train enters the port. The *GetTrain* process is shown in Figure A.7.

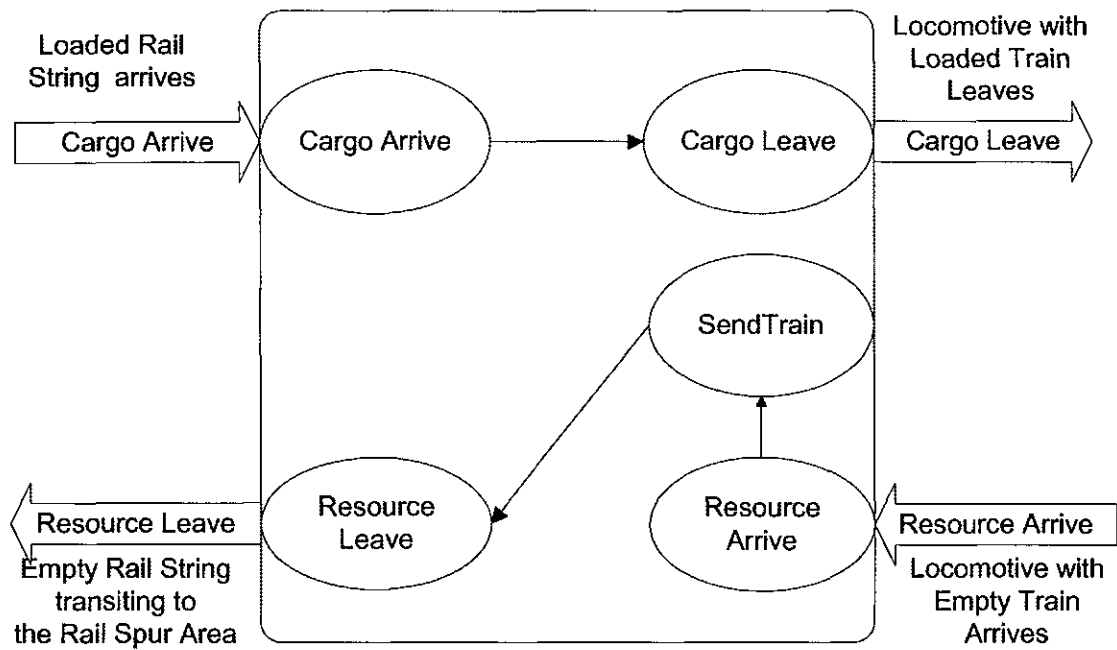


**Figure A.7 Model Flow Diagram for POD *GetTrain* process in an *InterchangeYardAreaSet* Object**

Before the train enters the port, it waits for the simulation to select an Interchange Yard Area (IYA) from the set of available IYAs. The selection is based on the maximum utilization of the available IYAs. If an IYA exists that can accommodate the train, then the IYA with the most available space is selected. If one is not available, then the train waits for an IYA to become available. The *SelectIYA* process in the *InterchangeYardAreaSet* object (Refer to Figure 4.2) is shown in Figure A.8. Once an IYA is selected, the train is triggered to enter the port. An IYA is shown in Figure A.9.



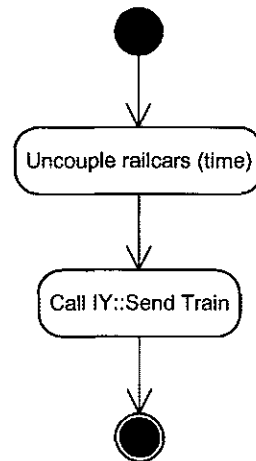
**Figure A.8 Model Flow Diagram for POD Select/YA process in an *InterchangeYardAreaSet* Object**



**Figure A.9 *InterchangeYardArea* Object**

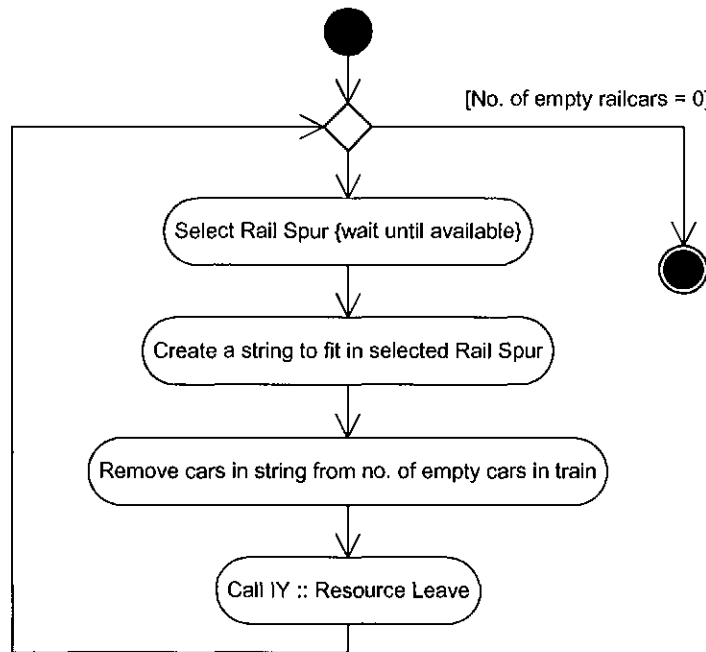
The *ResourceArrive* process in the *InterchangeYardArea* object (Refer to Figure 4.2) models the empty train entering the IYA. The rail cars of the train are uncoupled and the commercial locomotive that brought the train into the port is allowed to leave. The train is then ready to be sent to the Rail Spur Area (RSA) to be loaded with cargo. The *ResourceArrive* process is shown in Figure A.10.

Once the empty train has arrived in the IYA, the train is broken up into strings of rail cars to be sent to the RSA. It is possible that the entire train might not fit in the RSA at one time. Hence, the RSA that can fit the largest number of rail cars is selected. The string of rail cars are removed from the train and attached to a port locomotive which then transports it to the selected RSA.



**Figure A.10 Model Flow Diagram for POD *ResourceArrive* process in an *InterchangeYardArea* Object**

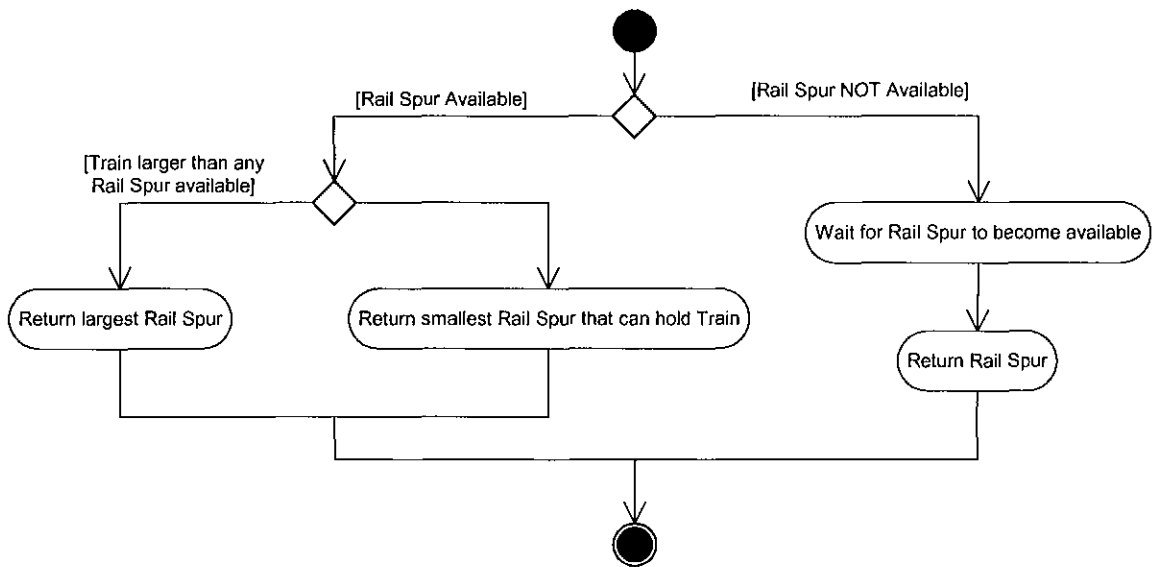
The *SendTrain* process in the *InterchangeYardArea* object (Refer to Figure 4.2) is shown in Figure A.11.



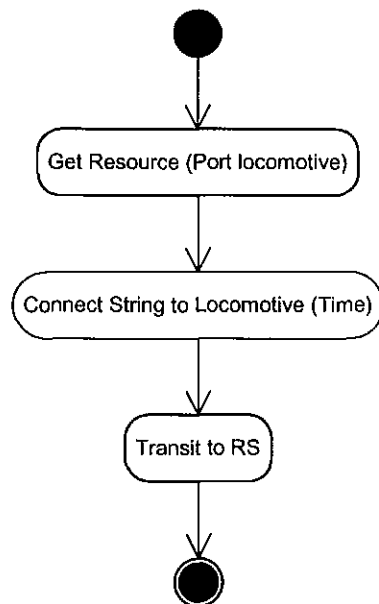
**Figure A.11 Model Flow Diagram for POD *SendTrain* process in an *InterchangeYardArea* Object**

If the entire train can fit into any of the available RSAs, then the train is directed to one of these RSAs. However, if the train is too large to fit into any of the available RSAs, then the train is broken up into strings of rail cars to fit into the available RSA. If no RSA is currently available, then the train waits for one to become available. The *SelectRSA* process in the *RailSpurAreaSet* object (Refer to Figure 4.2) is shown in Figure A.12.

Once an RSA has been selected, the string of rail cars waits for a port locomotive to arrive, which transports it to the RSA. The *ResourceLeave* process in the *InterchangeYardArea* object (Refer to Figure 4.2) is shown in Figure A.13.



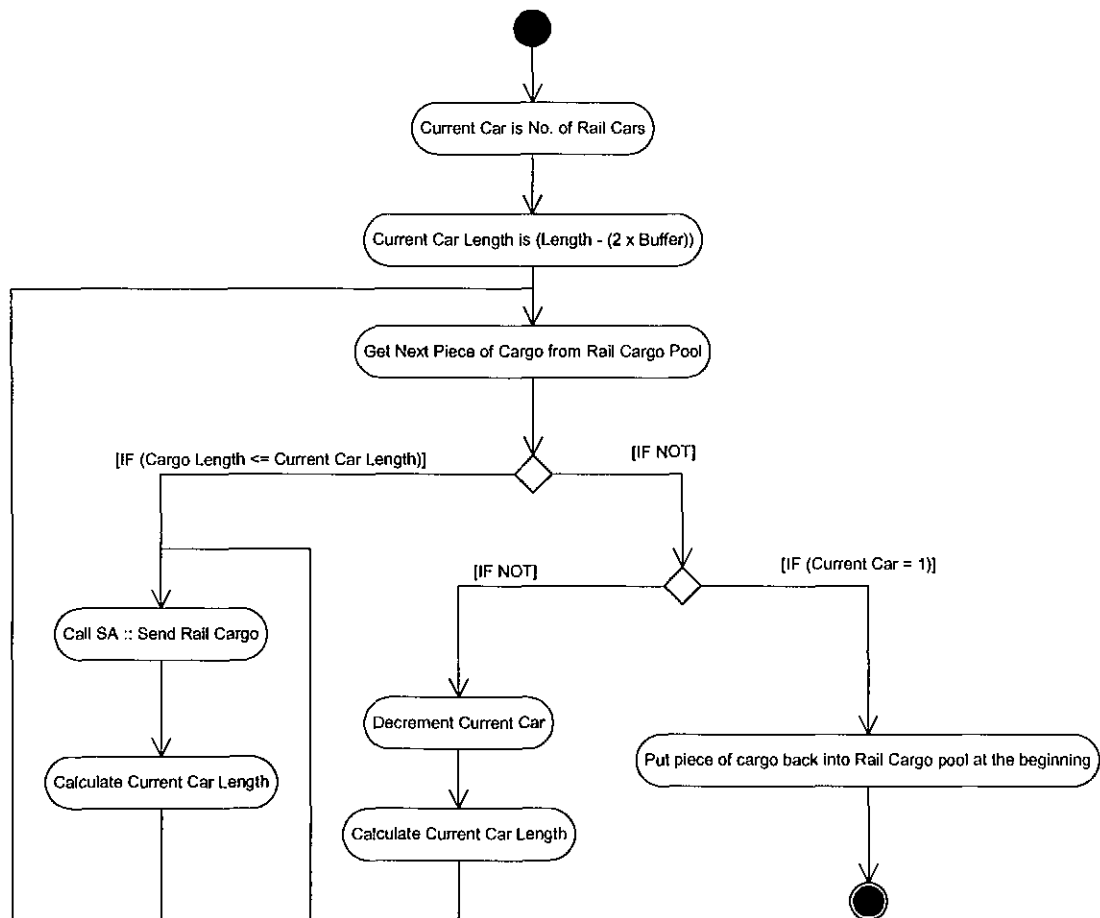
**Figure A.12 Model Flow Diagram for POD *SelectRSA* process in a *RailSpurAreaSet* Object**



**Figure A.13 Model Flow Diagram for POD *ResourceLeave* process in an *InterchangeYardArea* Object**

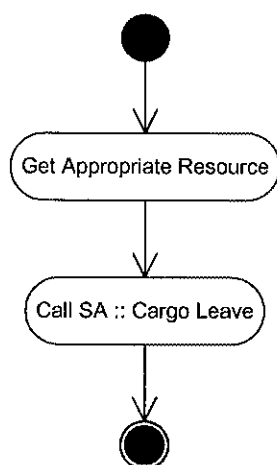
### A.3 Loading a train with its cargo

As the string of rail cars leaves the IYA for the RSA, it triggers the set of SAs to gather the rail cargo destined to leave on that string, and send it to the RSA for loading. The *GetRailCargo* process in the *StagingAreaSet* object (Refer to Figure 4.2) is shown in Figure A.14.



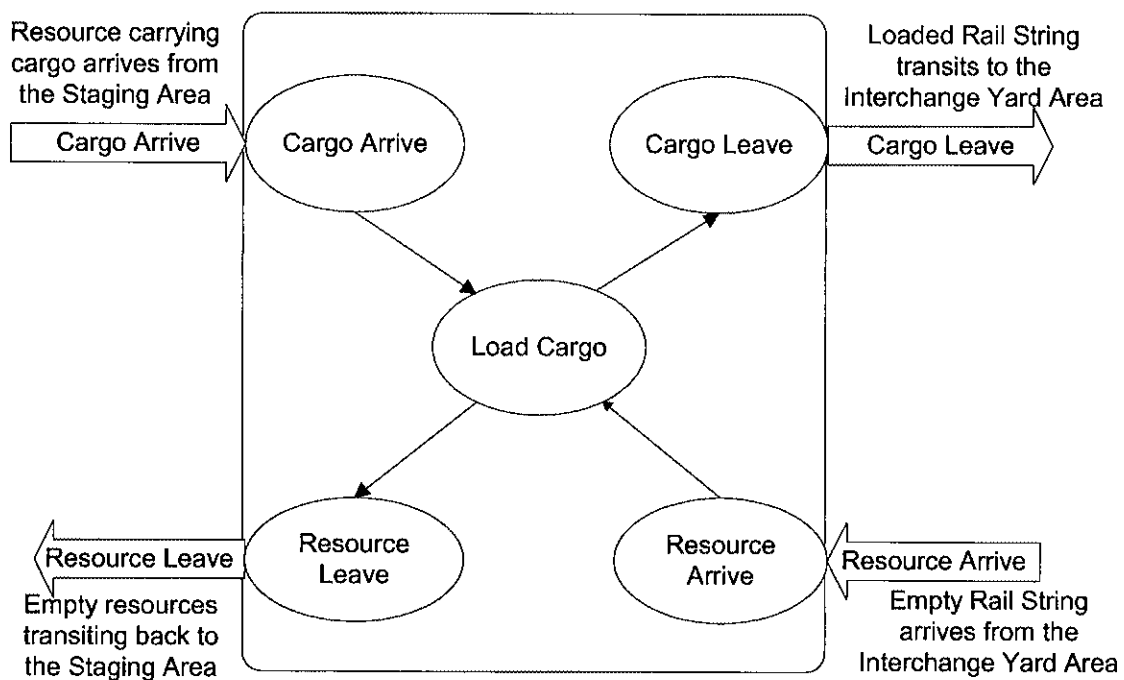
**Figure A.14 Model Flow Diagram for POD *GetRailCargo* process in a *StagingAreaSet* Object**

The individual SAs are triggered to send the cargo to the RSA via the *CargoLeave* process in the *StagingArea* object. The *SendRailCargo* process in the *StagingAreaSet* object (Refer to Figure 4.2) is shown in Figure A.15.



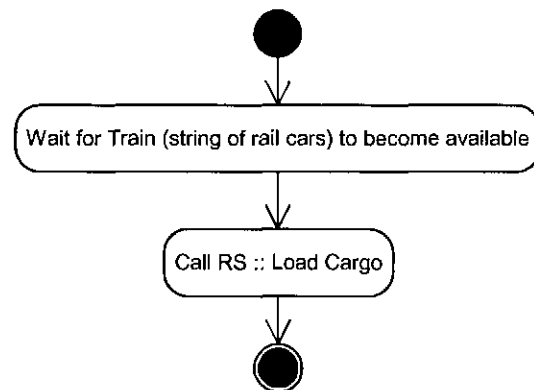
**Figure A.15 Model Flow Diagram for POD *SendRailCargo* process in a *StagingAreaSet* Object**

An RSA is shown in Figure A.16.



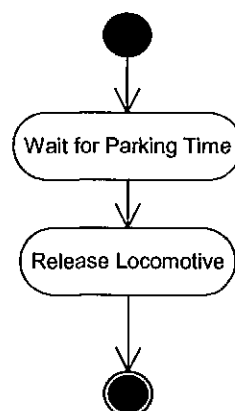
**Figure A.16 *RailSpurArea* Object**

Cargo scheduled to be loaded on to a train arrives in the RSA from the SA via the *CargoArrive* process in the *RailSpurArea* object (Refer to Figure 4.2). The cargo waits for the string of rail cars to arrive and is then loaded on to the train. The *CargoArrive* process is shown in Figure A.17.



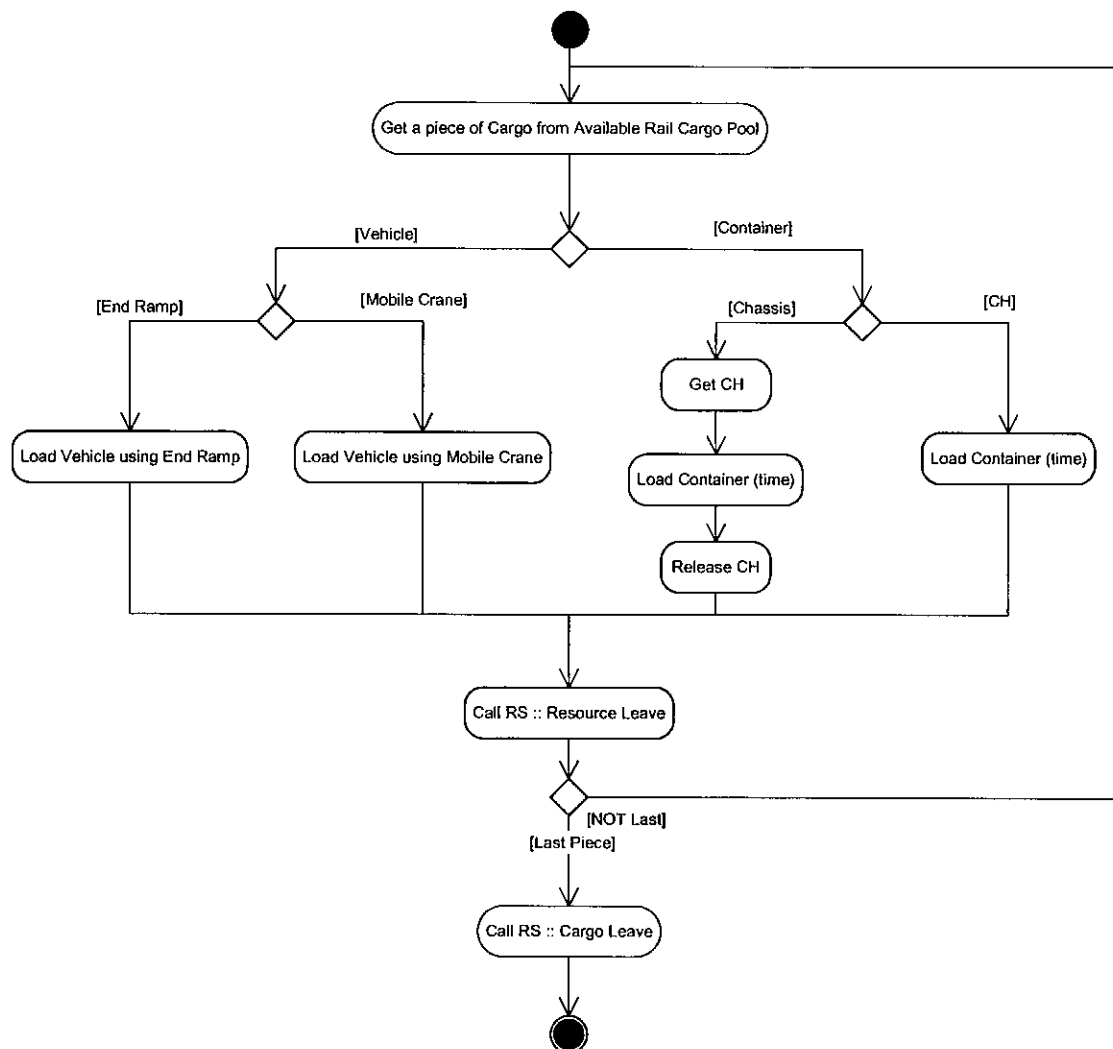
**Figure A.17 Model Flow Diagram for POD *CargoArrive* process in a *RailSpurArea* Object**

The string of empty rail cars enters the RSA via the *ResourceArrive* process in the *RailSpurArea* object (Refer to Figure 4.2), which is shown in Figure A.18.



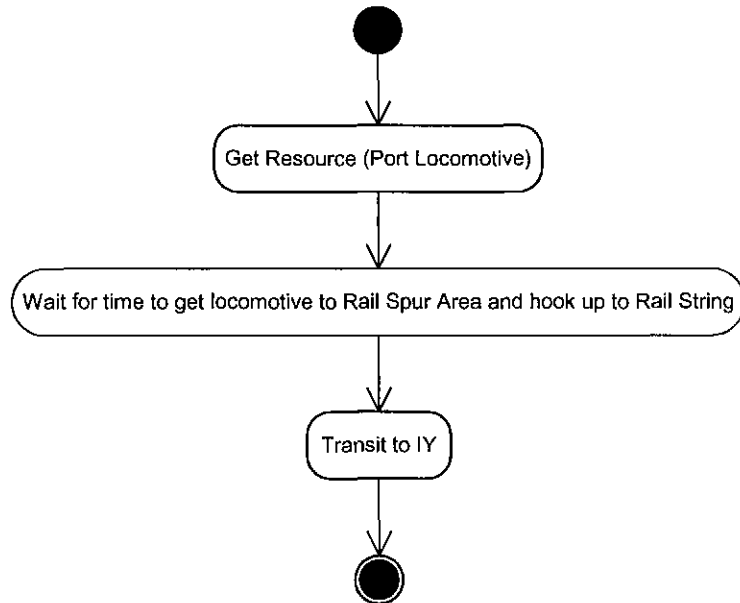
**Figure A.18 Model Flow Diagram for POD *ResourceArrive* process in a *RailSpurArea* Object**

Once the cargo as well as the string of rail cars to carry the cargo has arrived in the RSA, the cargo is loaded onto the rail cars. The process of loading the cargo varies depending on the type of cargo as well as the loading resources available in the particular RSA. The *LoadCargo* process in the *RailSpurArea* object (Refer to Figure 4.2) is shown in Figure A.19.



**Figure A.19 Model Flow Diagram for POD *LoadCargo* process in a *RailSpurArea* Object**

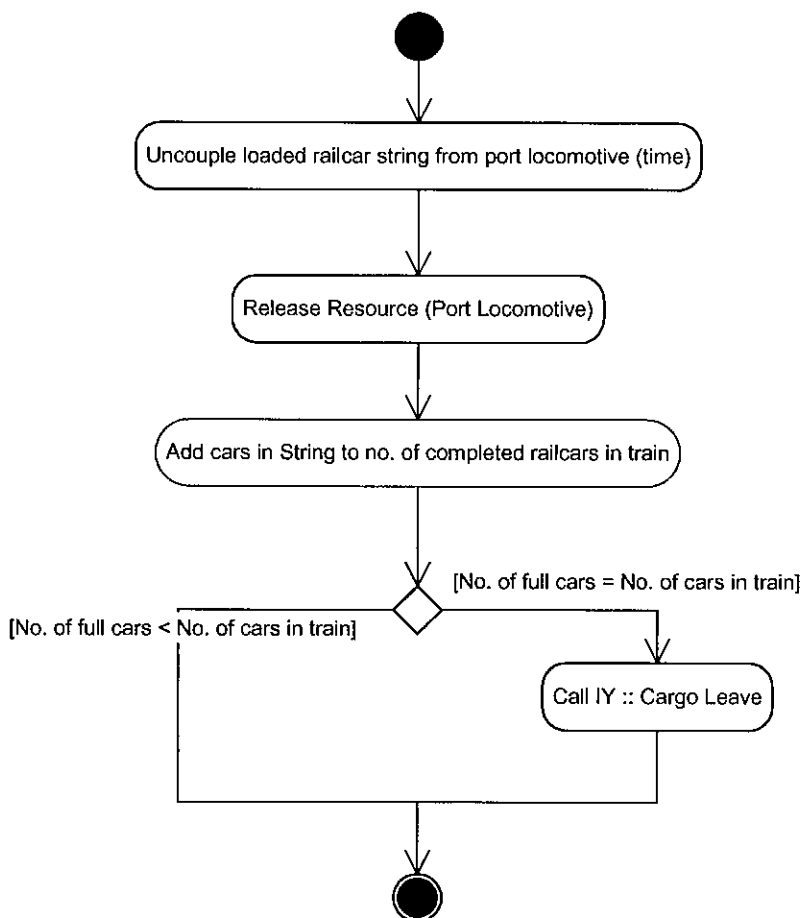
When all the cargo has been loaded on to the string of rail cars, the string waits for a port locomotive to pull it out of the RSA to the IYA where it came from. The *CargoLeave* process in the *RailSpurArea* object (Refer to Figure 4.2) is shown in Figure A.20.



**Figure A.20 Model Flow Diagram for POD *CargoLeave* process in a *RailSpurArea* Object**

#### **A.4 Train leaving the port with its cargo**

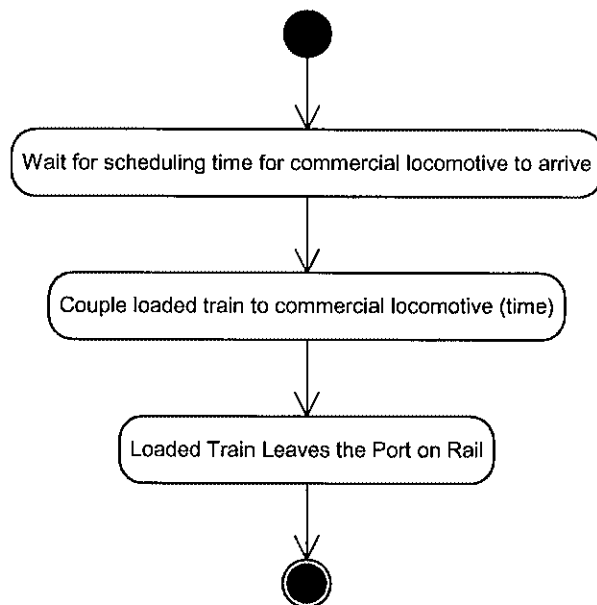
The *CargoArrive* process in the *InterchangeYardArea* object (Refer to Figure 4.2) is shown in Figure A.21. Once the string of rail cars arrives in the IYA loaded with cargo, the port locomotive is uncoupled from the string and the string is attached to the main train. If all the rail cars in the train have been loaded with their cargo, the train is sent out of the port via the *CargoLeave* process in the *InterchangeYardArea* object (Refer to Figure 4.2).



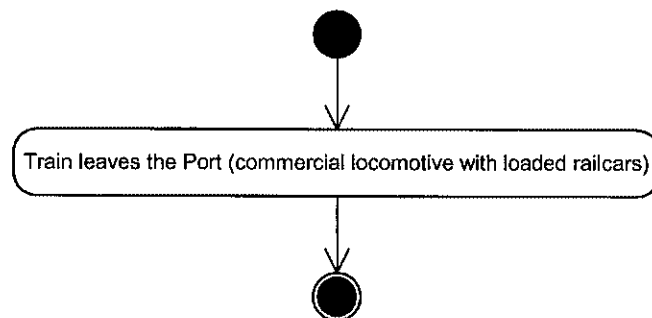
**Figure A.21 Model Flow Diagram for POD *CargoArrive* process in an *InterchangeYardArea* Object**

The train waits for a commercial locomotive to become available to haul it out of the port. The *CargoLeave* process is shown in Figure A.22. The train then leaves the port and travels to its destination.

The *TransportLeaveRail* process in the *Port* object (Refer to Figure 4.2) is shown in Figure A.23.



**Figure A.22 Model Flow Diagram for POD *CargoLeave* process in an *InterchangeYardArea* Object**



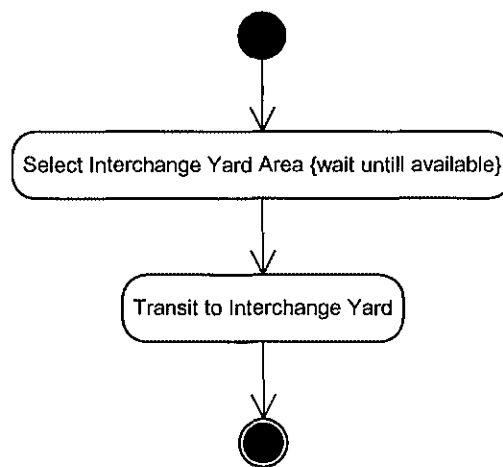
**Figure A.23 Model Flow Diagram for POD *TransportLeaveRail* process in a *Port* Object**

## **B. POE Rail Processes**

The processes associated with the clearance of POE rail cargo through a port can be described by following the path of a train arriving at the port loaded with cargo.

### B.1 Train with cargo arriving at the port

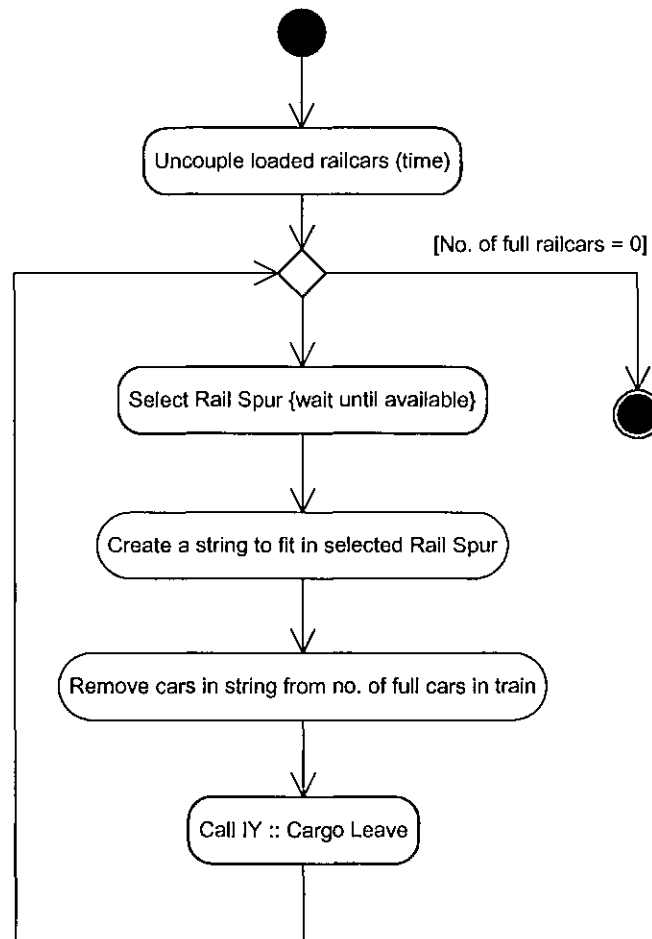
The POE rail cargo is brought into the port by trains arriving at the port with the cargo loaded onto them at a previous location such as an installation or a POD port. The trains wait for a suitable IYA to become available in the same manner as a POD train. As soon as one becomes available, the trains advance to the IYA. The *TransportArriveRail* process in the *Port* object (Refer to Figure 4.4) is shown in Figure B.1.



**Figure B.1 Model Flow Diagram for POE *TransportArriveRail* process in a *Port* Object**

### B.2 Train entering the port

In the IYA, the rail cars of the train are uncoupled and the commercial locomotive that brought the train into the port is allowed to leave. The train is then ready to be sent to the RSA to be unloaded. The train is split into strings of rail cars to fit the RSA. The *CargoArrive* process in the *InterchangeYardArea* object (Refer to Figure 4.4) is shown in Figure B.2.

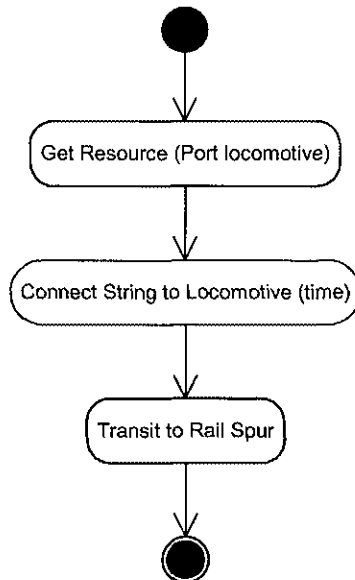


**Figure B.2 Model Flow Diagram for POE *CargoArrive* process in an *InterchangeYardArea* Object**

Once, the RSA has been selected, the string of loaded rail cars is sent to that particular RSA. The *CargoLeave* process in the *InterchangeYardArea* object (Refer to Figure 4.4) is shown in Figure B.3.

### **B.3 Unloading the train**

The *CargoArrive* process in the *RailSpurArea* object (Refer to Figure 4.4) is shown in Figure B.4. Once the string of rail cars loaded with cargo has arrived in the RSA, the cargo waits for a resource to become available to carry it to the SA.



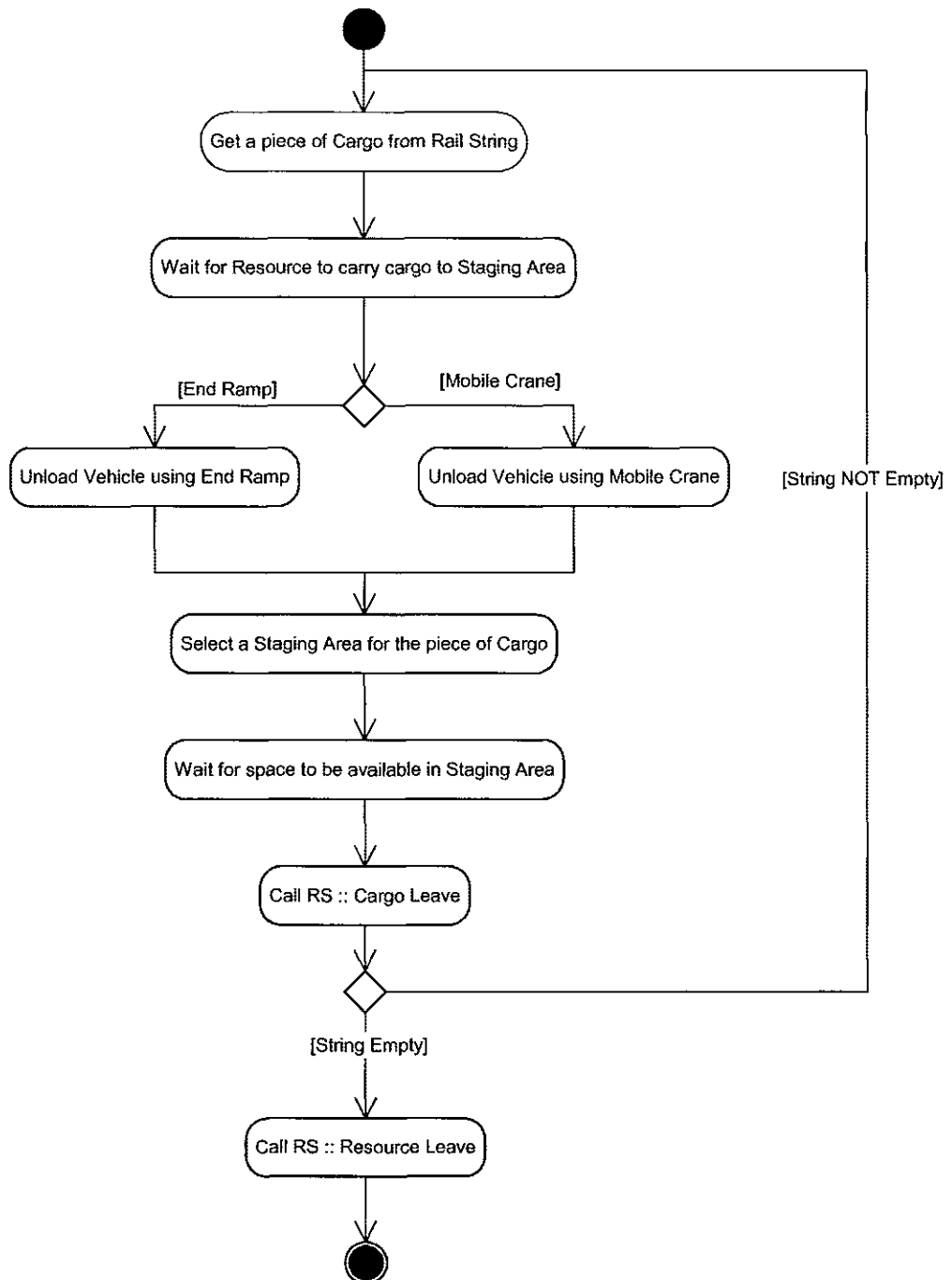
**Figure B.3 Model Flow Diagram for POE *CargoLeave* process in an *InterchangeYardArea* Object**

Once available, the cargo is unloaded and again waits for space to become available in a SA.

When space becomes available, the resource transports the cargo to the SA. The *CargoLeave* process in the *RailSpurArea* object (Refer to Figure 4.4) is shown in Figure B.5. This process continues until all the cargo has been unloaded from the string of rail cars. The string then returns to the IYA. The *ResourceLeave* process in the *RailSpurArea* object (Refer to Figure 4.4) is shown in Figure B.6.

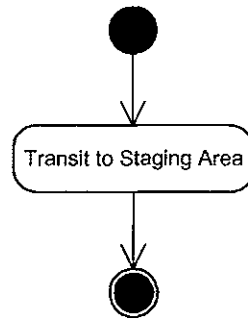
#### **B.4 Empty train leaving the port**

Once, the string of empty rail cars arrives in the IYA, the port locomotive is uncoupled from the string and the string is attached to the main train. The

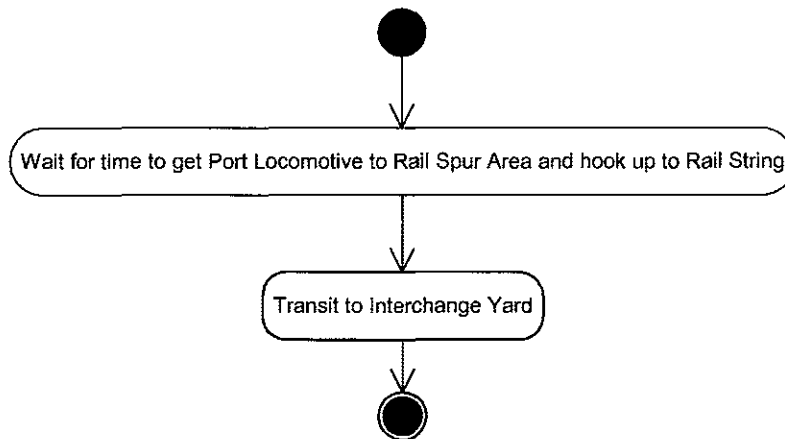


**Figure B.4 Model Flow Diagram for POE *CargoArrive* process in a *RailSpurArea* Object**

*ResourceArrive* process in the *InterchangeYardArea* object (Refer to Figure 4.4) is shown in Figure B.7.

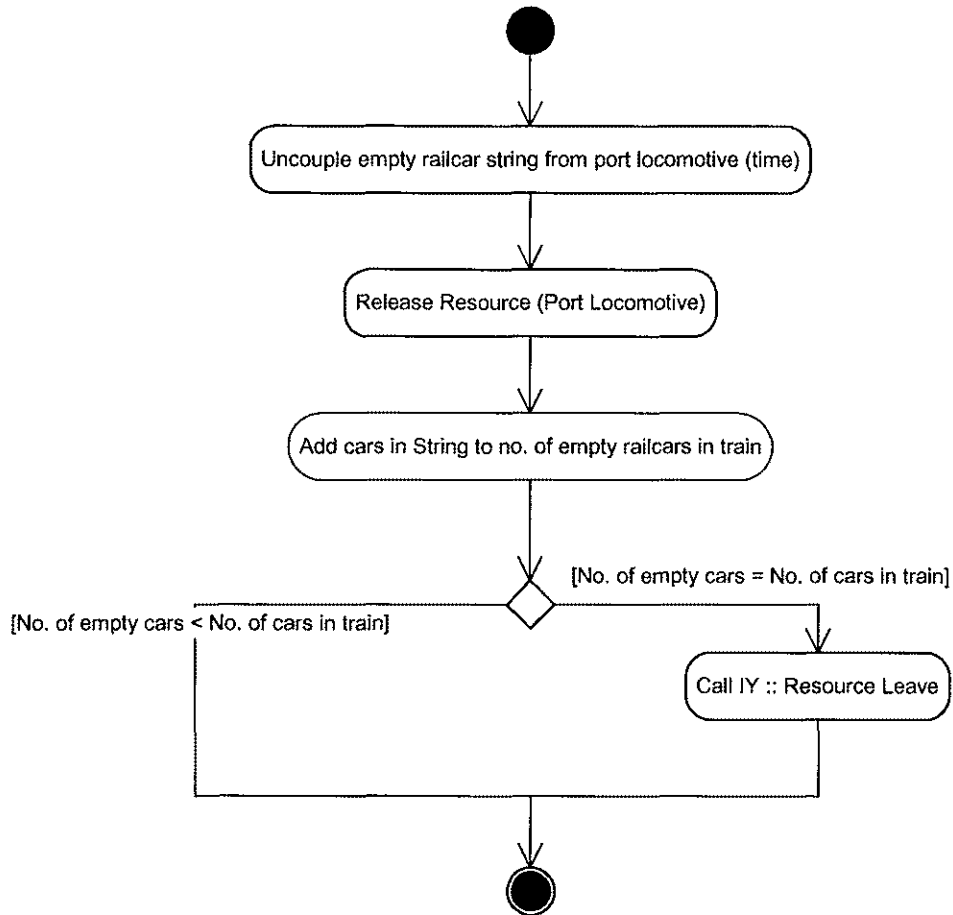


**Figure B.5 Model Flow Diagram for POE *CargoLeave* process in a *RailSpurArea* Object**

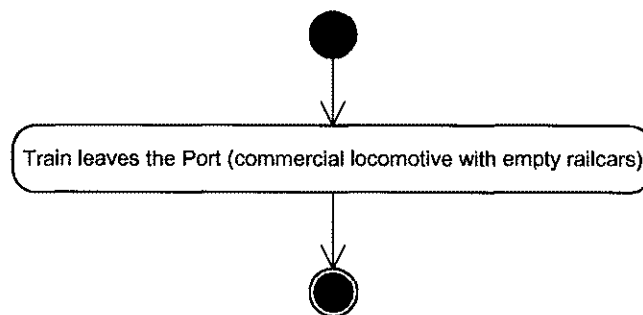


**Figure B.6 Model Flow Diagram for POE *ResourceLeave* process in a *RailSpurArea* Object**

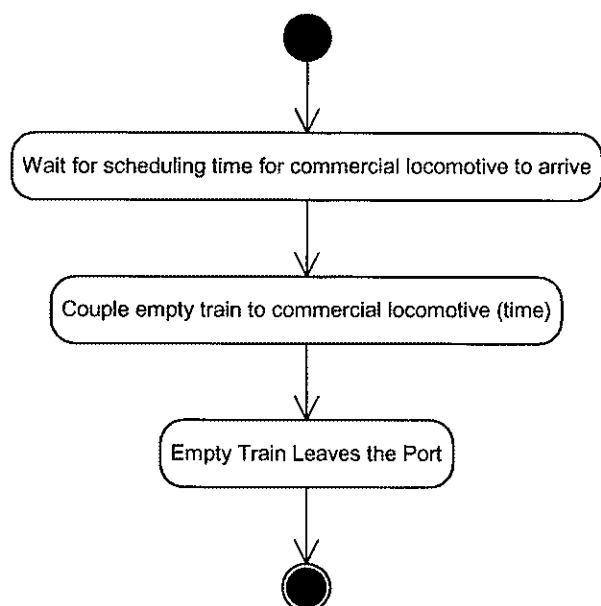
If all the rail cars in the train have been unloaded, the empty train is sent out of the port via the *ResourceLeave* process in the *InterchangeYardArea* object (Refer to Figure 4.4). The empty train waits for a commercial locomotive to become available to haul it out of the port. The *ResourceLeave* process is shown in Figure B.8. The *TransportLeaveRail* process in the *Port* object (Refer to Figure 4.4) is shown in Figure B.9.



**Figure B.7 Model Flow Diagram for POE *ResourceArrive* process in an *InterchangeYardArea* Object**



**Figure B.9 Model Flow Diagram for POE *TransportLeaveRail* process in a *Port* Object**



**Figure B.8 Model Flow Diagram for POE *ResourceLeave* process in an *InterchangeYardArea* Object**

**CURRICULUM VITA**  
**for**  
**REEJO MATHEW**

**DEGREES:**

Master of Science (Computer Engineering), Old Dominion University, Norfolk, VA, December 2002.

Bachelor of Engineering (Electronics Engineering), Sardar Patel College of Engineering, Mumbai University, Mumbai, India, June 2000.

**PART TIME EMPLOYMENT:**

Research Assistant at the Virginia Modeling Analysis and Simulation Center (VMASC) involved in the *PORTSIM (Port Simulation)*, *JLOTS (Joint Logistics Over The Shore)* and *Intra-Theater Sealift* projects, developing simulations of a port operation for the Military Traffic Management Command Transportation Engineering Agency (MTMCTEA) of the US Army. (August 2000 – Present)

Project Intern at the Tata Institute Of Fundamental Research (T.I.F.R), Mumbai, India in the Development and Analysis of a Receiver Circuit of a Zero Field Nuclear Magnetic Resonance (NMR) Spectrometer. (August 1999 – April 2000)

**SCHOLARLY ACTIVITIES COMPLETED:**

J. Leathrum, R. Mielke, T. Frith, R. Mathew. 2002. "Modeling New Technologies in a Joint Logistics Over The Shore (JLOTS) Operation," Summer Computer Simulation Conference (SCSC) 2002, San Diego, CA.