# A SIMULATION MODELLING PARADIGM FOR THE OPTIMAL MANAGEMENT OF LOGISTICS IN CONTAINER TERMINALS

Pasquale Legato and Roberto Trunfio
Department of Electronics, Computer Science and Systems
University of Calabria
Via P. Bucci, Cubo 42C, 87036 Rende (CS), Italy
E-mail: {legato, rtrunfio}@deis.unical.it

**KEYWORDS**
Modelling paradigm, discrete event simulation, transportation systems simulation and optimisation, container terminals.

## ABSTRACT

The need for an optimal management of logistic activities in modern container terminals is well recognized. Usually, the optimal management is pursued by decomposing the container terminal system in independent sub-systems to be optimized separately. Vice versa, a less practical approach, would suggest to analyze the whole system, provided that a formal, computerized model is available. We propose a simulation modelling paradigm based on the process interaction world view for the optimal management of logistic activities in a modern container terminal. Paper's focus is on the way of representing processes, resources and jobs that compose our simulation modelling paradigm. Processes are partitioned into activities following a simple to read and easy-to-use codified schema. We show how it is possible to define rules on the modelling paradigm for the optimization of the system. Concepts and representation methodologies are applied to a real case study, where it is asked to optimize the integrated management of berthing points, handling equipments and storage spaces.

## INTRODUCTION

Nowadays, approximately 90 percent of the world's cargo traffic moves by container (Sgouridis and Angelides 2002; OSC 2003; Notteboom 2004). An efficient and effective management of logistic activities in a container terminal can decrease the operating costs and service times and increase the quality of services. These results allow to achieve a better market position.

A container terminal is a complex system organised around a set of logistic processes. The logistic activities at a container terminal often belong to more logistic processes. This fact is critical for a good management of the system and the choice of the system modelling approach. Some interesting overview papers (Vis and De Koster 2003; Steenken et al. 2004) in the area of container logistics give a classification of the decision problems in a container terminal. According to these overview papers, the main problems in modern container terminals concern to the following processes: *i) arrival of the ship*, *ii) unloading and loading of the ship*, *iii) transport of containers from ship to stack and vice versa*, *iv) stacking of containers*, and *v) inter-terminal transport and other modes of transportation*.

Typical approaches for managing the problems in a container terminal are based on *reductionism* (such as DEVS formalism introduced by Zeigler (1976) and HCFG (Fritz and Sargent 1995)). Reductionism relies on the belief that a complex system may be decomposed into its constituent parts without any loss of information, predictive power or meaning (Pidd and Castro 1998). Unfortunately reductionism does not consider that a modern container terminal cannot be decomposed into its sub-systems without any loss of information, predictive power or meaning. This statement can be proved by trying to analyse the logistic processes separately: it is easily verified that they are partially or entirely related by means of logistic activities (Rizzoli et al. 1999). Pidd and Castro (op cit) have shown that the best approach for the management of a complex system is based on *wholism* (also known as *holism*). Wholism assumes that systems possess some properties that are meaningful only in the context of the whole and not in the parts. Therefore, using a wholistic approach, decision making in container terminal would need a mathematical model of the whole system.

A mathematical model can be examined by means of an analytical solution (as for an optimization problem) or by simulation. While optimization can allow the modelling of the whole system from a static, deterministic point of view, with a significant loss of information, simulation allows to keep a more realistic, dynamic and non deterministic, point of view on a complex system without loss of information. Therefore it should be preferable to develop decision support models for a container terminal that are based first on simulation and, after that, possibly also on embedded optimization models. Alternately, one stimulating possibility is that of using a simulator to pursue the so called "optimum seeking by simulation" (Law and Kelton 2000).

Using simulation, the first point to tackle with is the choice of the description language used to model the system. The logistic processes in container terminals classified by Vis and De Kostner (op cit) are usually modelled by queuing networks, due to the opportunity

Proceedings 21st European Conference on Modelling and Simulation
Ivan Zelinka, Zuzana Oplatková, Alessandra Orsoni ©ECMS 2007
ISBN 978-0-9553018-2-7 / ISBN 978-0-9553018-3-4 (CD)

of highlighting congestion phenomena occurring at those (shared) resources resulting as bottleneck within a logistic process. It is easily recognised that queuing networks do not offer a simple to read and easy-to-use modelling tool for high-complexity systems (Sauer et al. 1980). In a container terminal there is no clear distinction between resources and jobs involved in logistic processes, in the sense that one "entity" may act first as a resource and then as a job; moreover, multiple, complex rules of behaviour and operating policies are encountered in logistic processes, so the use of standard queuing networks does not appear to be effective. Therefore, a new wholistic simulation language for describing non standard queuing networks is necessary to model a complex system like a modern container terminal.

In this paper we propose a wholistic modelling paradigm (MP) for discrete-event simulation (DES) modelling based upon the process interaction (PI) world view. The remainder of this paper is organized as follows. In the next section we briefly describe the high-level architecture (HLA) of a tool for the optimal management of large and complex systems via DES. Afterwards we define the main concepts of our simulation MP and illustrate its potentiality by modelling some logistic processes at the Gioia Tauro maritime terminal.

## THE OPTIMAL MANAGEMENT OF COMPLEX SYSTEMS VIA DISCRETE EVENTS SIMULATION

We claim that a modern DES tool for the optimal management of large and complex systems must include *Simulation based Optimization* (SO) techniques. SO is the most important new simulation technology in the last decade (Law and McComas 2002). SO is the optimisation of performance measures based on outputs from stochastic (primarily discrete-event) simulations (Fu 2001; Ghiani et al. 2004). Law and Kelton (op cit) present a clear schema of the interaction between an optimization package and a simulation model (see Figure 1).

We propose the HLA of a modern DES tool for optimal complex system management (OCSM) by means of a *three phases approach*. The HLA implements the Law and Kelton SO schema (in fact in this schema, the simulation has a passive role in the optimization process). Our HLA is also valid for a continuous simulation tool, but we refer to DES because it is more appropriate for logistic processes.

### The Optimal Complex System Management Approach

The proposed three phases approach is made up of the following phases: *i) system modelling*, *ii) model analysis* and *iii) system optimization*. In the first phase, the real system is modelled using some MP, as discussed in-depth in the next section.

Numerical results from the simulation model obtained in the system modelling phase are analyzed in the second phase by means of a statistical analysis tool. Fundamental for the OCSM is the evaluation of performance measures. Performance measures are evaluated on the outputs from the simulation runs of the simulation model. In the model analysis phase, indices and parameters are defined on the simulation model by means of some tool or language. The evaluation of these indices and parameters provides the set of performance measures necessary to the OCSM.

After that, in the third phase an optimization problem is defined. General problem setting is made of input and output variables, objective function and constraints. The nature of the optimization problem is intrinsically stochastic, due to the nature of the output variables. Output variables are simulation model performance measures; input "variables" are quantitative or qualitative in nature. An example of qualitative variable is a queue policy.

Since the OCSM approach is essentially sequential, each phase of this management methodology is closely connected with the previous phases and involves the definition of a layered HLA for an OCSM tool.

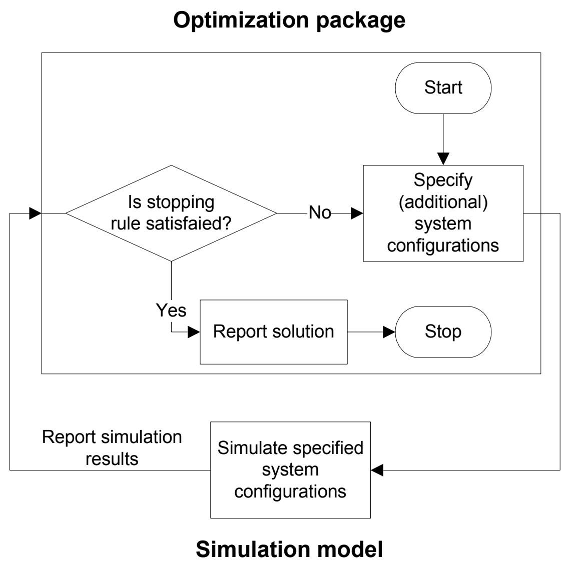

Optimization package
Start
Specify (additional) system configurations
Is stopping rule satisfaied?
No
Yes
Report solution
Stop
Report simulation results
Simulate specified system configurations
Simulation model

Figure 1: Interaction between an Optimization Package and Simulation Model

### The High-Level Architecture of the DES Tool

The HLA consists of three components, each related to one phase of the three phases proposed approach.

The base layer is the *simulation package* (SP). The SP is made up of the *simulation engine* and the *model builder*. The simulation engine provides the execution of simulation runs using such kind of simulation technique, e.g., batch means or independent replications (Nakayama 2002). The model builder is a tool for system modelling. It can implements *i)* a simulation language (e.g., GPSS, Slam), *ii)* a MP (e.g., HCFG, DEVS) or a VIMS (e.g., Arena, exteNd).

The intermediate layer is the *analysis package* (AP). The AP is composed by: *i)* a tool for the definition of indices and parameters on the simulation model (the *data definition builder*); *ii)* the *evaluation engine*, that interacts with the simulation engine to collect and evaluate simulation data and for computing the performance measures; iii) the *model analyzer*, that allows to manage the system by means of the analysis of the performance measures.

The higher level is the *optimization package* (OP). The OP must be more general as possible, because optimum-seeking can be obtained in a lot of way (Fu 2001; Ghiani et al. 2004). Our concept of an OP is of a building without walls: the modeller build the walls in dependency of the problem. So an OP must provides these functionality: *i)* executing and stopping simulation; *ii)* retrieval of data from simulations and model analysis; *iii)* designing of optimization algorithms; *iv)* definition of rules that allow an optimization algorithm to generate a new feasible system configuration.

In Figure 2 the dependency between the HLA components described above is illustrated.

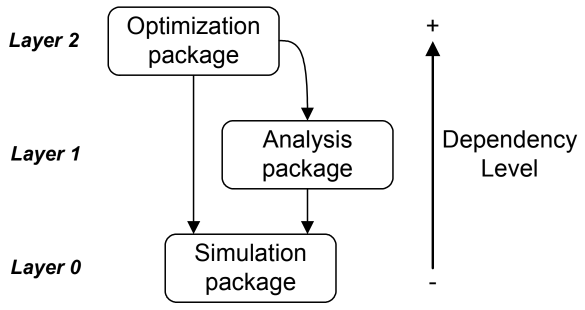


Figure 2: Dependencies in a High-Level Architecture of a Modern Tool for the Optimal Management of Large and Complex Systems

## A WHOLISTIC MODELLING PARADIGM

This section is devoted to explain why a PI world view is adopted at the basis of the wholistic MP. A conceptual framework CF (or world view) is an underlying structure and organization of ideas which constitute the outline and basic frame that guide a modeller in representing a system in the form of a model (Derrick et al. 1989).

### An Overview of the Process Interaction World View

PI world view is widely used in many MPs to model large and complex systems; sometimes is used a modified version of this CF, e.g., the HCFG developed by Fritz and Sargent (1995) uses the *modified process world view* (Cota and Sargent 1992). The reason of this success is explained by the following important features: *i)* a moderate *burden on modeller*, *ii)* an excellent *natural representation capability* and *iii)* a high *model maintainability* (Derrick, et al. 1989). The most important drawback of this CF is the high *burden on executive*, but it could be overcome with parallel or ditributed applications.

In the PI world view, each process routine in a model specification describes the progress of a particular model object through a series of time related activities (Overstreet and Nance 2004). So, the first thing to do in the PI world view is to identify all model objects whose activity sequences must be defined. Subsequently, the modeller specifies the sequence of activities for each object. For this reason, often a process is represented with a schematic representation known as *flow-chart*, where the activities that compose the process routine are nodes of the chart. The flow-chart are usually represented as a single source multi-sink directed graph (Silver et al. 2006). For each activity into a process routine there is a stretch of simulation execution. An activity obtains the *control* by a preceding activity, until the control is transferred to a subsequent activity. Control may be transferred in two ways: *i)* a definite delay may be handled by scheduling its next reactivation on the future activation list at a particular time in the future, or *ii)* an indefinite delay may also be used, in which case the process becomes indefinitely suspended until something outside this process happens (e.g., a condition becomes true).

### Wholistic Discrete Event Simulation Models with Process Interaction Modelling

The MP proposed in this paper is aimed to be flexible and expressive in the modelling of complex systems. It seeks to achieve three primary objectives: *model readability*, *reusability* and *customizability*. Model readability is that property which allows a model to be simple-to-read for a non-modeller. Here this objective is achieved by describing a simulation model by a flow-chart.

A need in modern simulation tools is reuse in some of its forms (model reuse, component reuse, function reuse and code scavenging) (Pidd 2002). According to our concept of wholistic MP, we provide model reusability by means of hierarchical model definition and a lot of parameters redefinition.

Model customizability is the base of a MP for the modelling of complex systems. It relies on the user-definition of process properties that allow to describe uncommon situations.

An outline of the MP follows now. A simulation model includes a set of model *objects*. For each model object a sequence of activities and events is defined. The sequence is also called the *object process routine*, or simply *process*. A process is represented as a *hierarchical flow-chart* (HFC), where activities and events are nodes and edges fix the logical sequence between nodes. Activities can be simple or composite. A *simple activity* is described by a simulation quasi-language; a *compound activity* is made of two or more activities. Indeed, a compound activity is a HFC but it is not meaningful as a process.

A simulation model can also include one or more *sub-models*, also called *sub-systems*. For each sub-model, a model object (so a process) manage the interaction between the sub-model and the objects in the model. Objects that belong to a model have no visibility of the outside, but they can widely see into the model and sub-models. This feature helps in model reuse and is not a violation of the wholistic approach: in the next paragraph it will be shown how to overcome this restriction.

Similar to the conical methodology (Nance 1987; Derrick et al. 1989), a model definition and specification using three main tools is provided.

The first tool is the *model hierarchy structure* (MHS). The MHS is a tree that provides an high-level model definition showing hierarchically the processes (so the objects) and sub-models that compose a simulation model (see Figure 3 for an example).

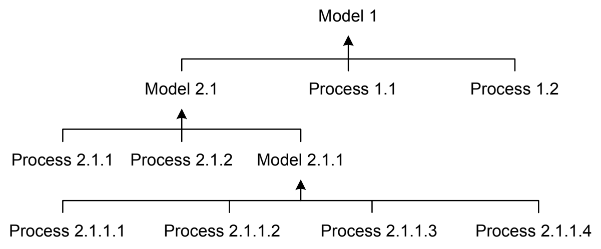


Figure 3: A Model Hierarchy Structure (MHS)

A more detailed model definition is obtained via the second tool, the HFCs. A HFC provides the process definition. In fact a HFC define the structure of a process by means of a flow-chart. Because of the possible use of composite activities for the process definition, a HFC could be a hierarchical structure (composite nodes are expanded revealing a new HFC).

The last tool is a simulation quasi-language (Pidd 2004) that provides the simple activities and events specification. It can be seen also as a tool for the process specification. Indeed, one could merge the events and activities specification obtaining a process specification, but in this way the readability of the model marked by HFCs would get lost.

## Model Objects

Following the PI world view, the first step is to identify all the model objects which are involved in the system modelling and to detect all common features. Then class-dependent features are described for the few classes of model objects introduced in our MP. The objects are contextual, so one need is to specify the model in which they are defined. The *model name* parameter is used to declare which model an object belongs to. This parameter is necessary for two reasons: *i)* the wholistic approach says that such objects can be used only in some contexts; *ii)* the use of sub-models could infers a lot of confusion (e.g., the same object used in a model and in its sub-models). The model objects of the same type are identified by the *type name* parameter. If the model name is a parameter that depends from the model, the type name is a non-changeable parameter. Each instance of a certain type of object is also identified by the *instance name* parameter. The set composed by these three parameters univocally identifies a model object.

There is at least another important parameter that allows to manage heterogeneous objects, known as *category name*. The use of the category name allow us to make associations between objects that are apparently disjoined. A possibile use can be seen in the AP and OP of a tool for the OCSM, where generic rules and algorithms can be defined over a class of mixed objects.

Model objects are illustrated in detail here.

There are two basic classes of objects: *i) resources* and *ii) jobs*. Both resources and jobs are *stateful*; besides, they are *active* or *passive* in function of their role in the simulation model.

An *Active resource* can possess other resources and it can offer a service at one or more jobs per time. It can also make queries to other objects. *Passive resources* can be possessed by other objects following a customized path (indeed, the activity sequence that compose the resource process routine). We indentify five states for an active resource (see Figure 4 for the active resource state diagram): *i) unavailable* – the resource is prevented from servicing requests (e.g., it is broken or locked ); *ii) idle* – the resource is available for servicing and is waiting for a request to be serviced; *iii) busy* – whenever the resource is servicing a request; *iv) hold* – whenever the resource is blocked/suspended and is waiting for something (e.g., it needs a tool to complete a service); and *v) in evaluation* – if the resource is doing the special activity consisting in taking decisions (e.g., determine which request to execute next, or which resource to visit next (by a job)).

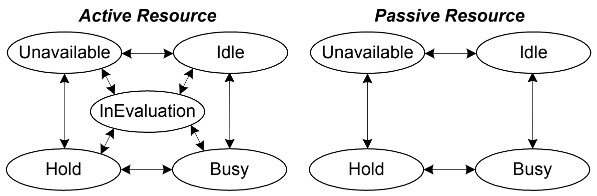


Figure 4: Active and Passive Resources State-Diagram

The state diagram for a passive resource (Figure 4) differs from the active resource state diagram for the lack of the “in evaluation” state. In fact, a passive resource has no capability to take any kind of decision, because of its passive role in the simulation model.

*Active jobs* can take a decision, they can call for a service and they are able to take and hold a passive resource and other jobs. As active resources, they can make queries to other objects. *Passive jobs* are only able to follow a certain path, described by the their own process routine. The states for an active job are four: *i) incoming* – the job is arriving from outside and defining the next service request (e.g., a ship that is travelling to the terminal); *ii) in evaluation* – the job is doing an activity aimed to take a decision (e.g., it is asking/negotiating for a service by a resource, it is thinking whether (or not) it has to abandon the queue); *iii) in queue* – it is waiting for a service; and *iv) in service* – the job is under service. The active job state diagram is shown in the Figure 5.

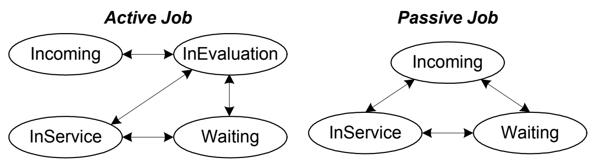


Figure 5: Active and Passive Jobs State-Diagram

Clearly, a passive job (e.g., a container) does not have any “in evaluation” state.

With the basic objects introduced up to this point one could represent, in an effective way, just a system governed by a few simple rules.

As matter of fact, the need of modelling complex systems lead us to introduce another class of model objects, the *resource managers*. A resource manager is an high-level object, that is able to interact with a set of model objects. The set of model objects can also include objects that belong to sub-models. Resource managers can take decisions (e.g., solve a scheduling or an assigment problem, negotiating the use of a sub-system, etc.) applying rules and policies and making queries to other model objects. They have free access to modify the behaviour of all the resources into their own model and included sub-models.

The resource manager is an atypic model object in nature. In fact, unlike other model objects, resource managers are stateless and they are only active. However their process can be stopped by others resource managers from the same model or upper.

## Processes and Hierarchical Flow-Chart

The role of processes in model definition is analysed here and process representation is shown. Process classification is also given and the interaction of processes and model objects during the simulation is described.

According to the definition of process given in the PI world view overview, a process is a series of temporally related events and activities. Flow-charts are used to represents processes. Events and activities are nodes, while edges define one or more paths that the process can start. Others elements compose a process: *transition points* and *logical nodes*.

Processes are *stateful* and they are usually executed during more simulation time periods. For this reason the nodes of a process can be visited during different time periods. To track this possibility, we adopt the *process checkpoint*. A process checkpoint is a property of the processes that shows the node from which a process starts or it is reactivated. A process checkpoint is also called *reactivation point*.

A process can be in one out of four possible states (Dahl and Nygaard 1966): *active*, *suspended*, *passive* and *terminated* (see Figure 6). Focusing on a non-concurrent simulation, the meaning of the states can be explained as follows.

A process is active in the sense that it is performing its activities and scheduling its events until it enters one of the other three states. A suspended process has a process checkpoint and it is waiting for a certain event notice; unless a change of state is caused by another process, the suspended process will be again active when the event notice becomes the current one. The passive state also has a process checkpoint, but no associated event notice. A process will remain passive until another process will change its state. Finally, a process becomes terminated whenever control passes through a termination node or the process is definitely interrupted by another one.

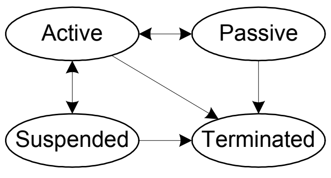


Figure 6: Process State Diagram

The graphical representation of a process by a HFC allow us to achieve at a good extent the readability objective. Reminding the list of the process components, let us start an in-depth discussion about these components.

In our MP two kind of activities classification are used. The first classification focuses on the simulation duration of the activity, therefore activities are partitioned in *timed* and *instantaneous*. The first type of activities are those that start operations at a simulated time instant and finish operations in a future simulated time instant. The second type refers to activities that starts and ends operations at the same simulated time instant.

The second classification classifies four type of activities: *operative*, *logic-based* and *decision-making* (see Figure 7). The last type is a mix of the previous activities and is called *compound*. The operative activities are those that perform operations characterized by a variable simulated time length (e.g., a service time, a waiting time). The logic-based activities are those that perform operations whose simulated time is equal to zero (e.g., a service request). The decision-making activities are those activities that make some choice using policies, rules or algorithms. These kind of activities, as the logic activities are timeless. On the contrary, operative activities are time-consuming.

The compound activities can belong to both timed and instantaneous activities, depending on the composing activities. As process, composite activities are defined using a HFC; if a process that belongs from a submodel is an open-process it can compose a compound activity. Also for this reason, a compound activity can contain an event.

An event is a fact that forces one out of a set of possible changes of the current state. It may precede or follow a timed or istantaneous activity, thus representing something that is just happened or that is going to happen. If an event precedes an activity, then it is processed at the same simulated time of the activity start; if an event follows an activity, then it is processed at the same simulated time of the activity end.

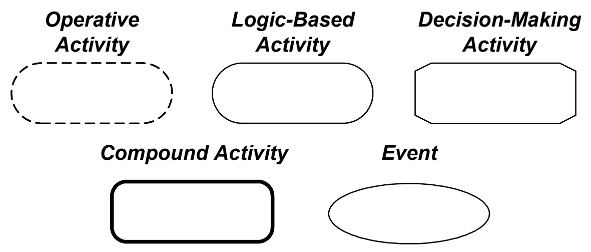


Figure 7: HFC Activities and Events

Our MP allows to define a process via HFC using activities, events and others elements as *transition points* and *logical nodes*.

Transition points, help in definition of non-cyclical and open process, and all the composite activities. They are the *starting point* and *ending point* (Figure 8). A process could have none or one starting point and none, one or more ending points. If the process checkpoint is on the starting point, it has the active state; otherwise, if the process checkpoint is on an ending point the process is terminated and it will never be re-used.

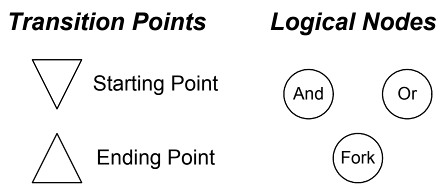


Figure 8: HFC Activities and Events

Logical nodes support definition of process paths. They are the *and*, *or* and *fork* nodes (Figure 8). A logical node may be used to represent alternative paths to be chosen under specified conditions.

Another important classification looks at the process from the model object activities sequence that it describes. We distinguish in *job process* (JP), *resource process* (RP) and *resource-manager process* (RMP). A process do not interact with other processes, because it is only a picture of the role of an object in a model. A process only interacts with its own model object. So, if there are two objects, *A* and *B*, the related processes can only interact with their respective model objects. Vice versa, the model object A can directly interact with B following the process A instructions. The object B will answer A by means of the process B instructions.

## A Simulation Quasi-Language

The model specification is obtained via the HFC components specification. A simulation quasi-language (Pidd 2004) has been developed for the specification of HFC components, to achieve the objective of model customizability. The benefit of using a simulation quasi-language for *system definition* has been recently shown in a context of production logistics (Schulze et al. 2000). The high level of detail gained in model specification is very important, in practice, for system understanding and for getting a simple-to-write/read model also to a non-modeller. In our case study, the description of a set of complex policies and rules for resources allocation and activities scheduling at a container terminal involves both simulationists and heads of various operative areas, that are required to work jointly and to fix modelling agreements.

Our simulation quasi-language is a descriptive language to write a process using a human-like language. It has no explicit reference to the used parameters (e.g., the distribution used for the generation of a service), but it describes the system behaviour clearly. The task of simulation parameters specification is reassigned to a low level language, because these parameters are meaningful to the modeller only.

```
LogicBased Activity "Service Request"
   Job InEvaluation;
   Job Requests Server to ServerManager;
   Starting of Activity "Verification of
   Server Availability";
End Activity
DecisionMaking Activity "Verification of
Server Availability"
   If Server is not Available
      Activity "Waiting for Server" starts;
   Else
      Activity "Service" starts;
End Activity
Operative Activity "Waiting for Server"
   Job Waiting until Job Obtains its Turn;
   Activity "Service" starts when Server
   Starts Service;
End Activity
Operative Activity "Service"
   Job InService until Server Ends Service;
   Event "Job Service Ends" Starts
   Goto ExitPoint
End Activity
```

Figure 9: A Job Requests a Resource

For each process is necessary to completely describe each HFC component. The language works with blocks. We have six types of blocks: `Define`, `Rule`, `Activity`, `Event`, `TransitionPoint` and `LogicalNode`. The first and second block are optional, whereas the others depend on the HFC composition. Basic elements of the language are terms like `Job`, `Resource`, all the possible model objects states (e.g., `Hold`, `Waiting`) and `System` for sub-models.

An example of a job process specification is shown in Figure 9.

## A REAL CASE STUDY: THE GIOIA TAURO TERMINAL

In a previous research on simulating container terminals, it was conducted a scenario analysis and overall performance evaluation (Legato and Mazza 2001). The strategic nature of the problem at hand, as well as the "standard" performance indices expected in output lead us to use a commercial simulation package – Pritsker's *AweSim*. Our following work (Canonaco et al. 2007) focused on a more operational issue: the optimal management of container discharge/loading at any given berthing point. We carried out some *what-if* attempts logically defined by a manually performed local search on a few neighbor configurations of the discharge/loading schedule set by the current practice. In order to work at this "lower level", we implemented the simulation model with Borland's *Delphi* SDK. As a result of our past experience, today we face the opportunity-necessity of defining and using a context specific MP-based environment for the optimal management of logistic activities in modern container terminals.

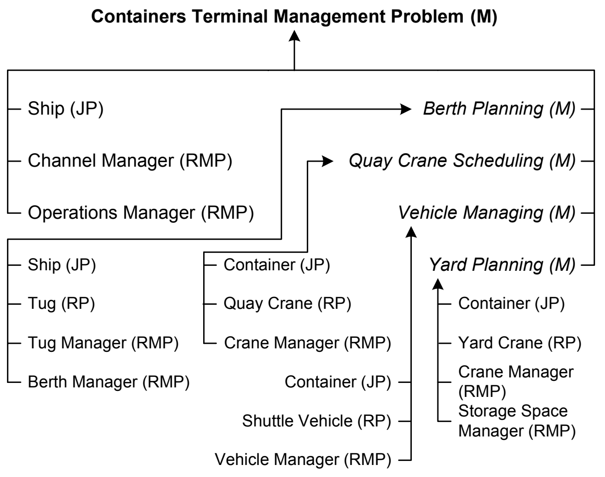


Figure 10: MHS of the Container Terminal Management Problem

A major reason that makes appealing the wholistic approach to processes organization consists in the possibility of replicating the company, typical organization chart for managing the complete set of logistic processes identified. In particular, wholism seems the most appropriate paradigm for supporting the "whole vision" used by the Terminal (operative) Manager in getting a "whole optimal objective". This Manager is faced with the composition/modification of the partially optimal decisions and procedures adopted by a set of competitive actors which are in charge of managing each subsystem (berthing points, cranes, storage yard and shuttle vehicles). Thus, he has to revise objectives, to introduce constraints and so on. In one word, as prerequisite of its action he needs "to know what".

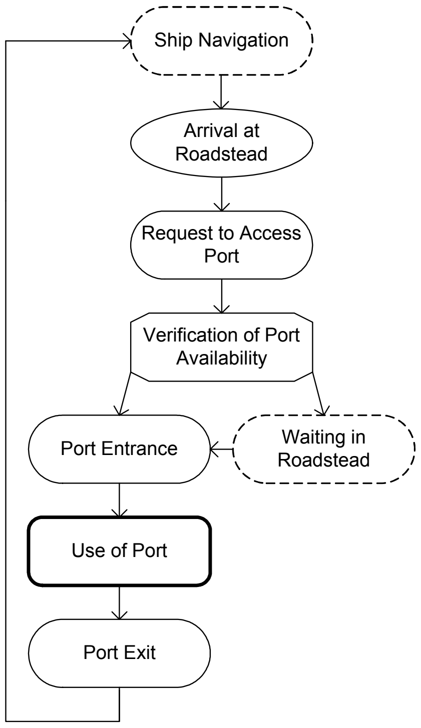
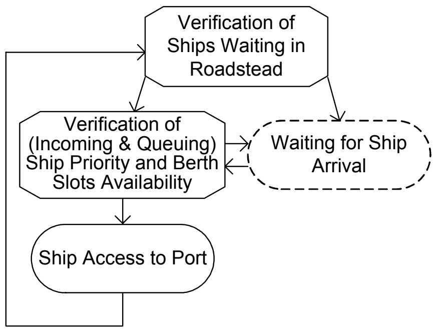


Figure 11: HFC for Ship Process



Verification of Ships Waiting in Roadstead
Verification of (Incoming & Queuing) Ship Priority and Berth Slots Availability
Waiting for Ship Arrival
Ship Access to Port

Figure 12: HFC for Channel Manager Process

The organization of all the processes and models into a "whole" hierarchical system for terminal management is resumed in Figure 10. In particular, one may see the typical basic models (*Berth Planning*, *Quay Crane Scheduling*, *Vehicle Managing* and *Yard Planning*) around which all the processes are grouped. The wholistic approach reveals its effectiveness to answer the need of "to know what". This statement primarily means that for an active object the decision making needs to know the behaviour of others objects. This fact is especially true for resource managers (as the

Operations Manager), that can interact and act on other model objects.

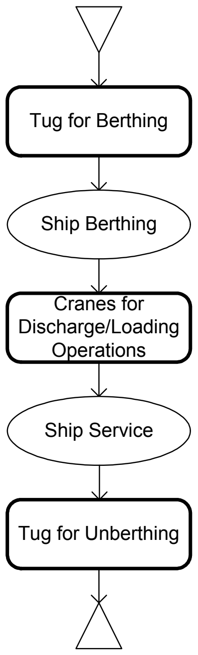
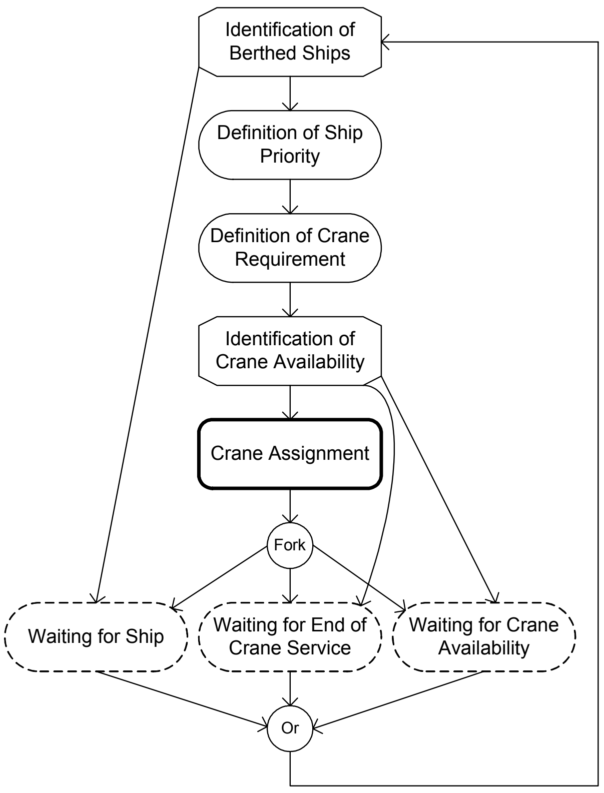


Figure 13: HFC for Ship Process into the Berth Planning

For sake of completeness, the MP based representation of the real logistic process used in the following for illustrative purposes will be preceded by a short description. When calling at a port, a ship makes a sort of "advanced reservation" based on its ETA - Expected Time of Arrival. Ship entrance depends on formal conditions (e.g. contractual agreements between the ship's shipping line and the port of call for the use of port facilities), as well as operational settings (e.g. berth assignment and pilot/tug availability). If requirements are met, the ship get the resource "channel" and is maneuvered down the navigation channel and into its berth slot by one or more tugs; otherwise it must wait outside, in the roadstead. This initial part of the process involves both the berth and the channel manager.

Once the ship is berthed, container discharge/loading can be initiated only if additional human and mechanical resources are allocated; if not, the ship waits in its berth position until resource assignment. In particular, at the Gioia Tauro terminal, discharge/loading operations are performed by rail mounted gantry cranes placed along the berth: one or multiple cranes move containers between the ship and the berth area. When multiple cranes are assigned to the same ship, crane interference has to be avoided and a complex scheduling problem arises to manage the relationships (precedence and mutual exclusion) existing among the holds of the same ship. A fleet of shuttle vehicles take in charge containers and cycle between the berth area and the assigned storage positions within the yard. A key problem for the manager of the fleet is the allocation of the optimal number of shuttle vehicles to each couple of pick-up and delivery points as well as the determination of the related optimal paths.



Identification of Berthed Ships
Definition of Ship Priority
Definition of Crane Requirement
Identification of Crane Availability
Crane Assignment
Fork
Waiting for Ship
Waiting for End of Crane Service
Waiting for Crane Availability
Or

Figure 14: HFC for Berth Manager Process

As soon as the above operations are completed, the ship requests, and eventually waits for, tug services to un-berth and exit the port via the navigation channel. The return of the ship to the roadstead definitely closes the set of logistic processes: *i) arrival of the ship*, *ii) unloading and loading of the ship*, *iii) transport of containers from ship to stack and vice versa*, *iv) stacking of containers* and *v) departure of the ship*.

With reference to the "arrival of the ship" process described, we identify two model objects: *ship* and *channel manager*. The former is a job, whose lifecycle is primarily defined within the *berth planning* model; the latter is a resource manager, that decides whether (or not) to admit a ship to the port on the basis of some priority mechanism. The HFC representation of processes for both model objects is given in Figures 11, 12 and 13.

As for the remaining logistic processes, we concentrate on two model objects: *berth manager* and *shuttle vehicle*. The berth manager assigns cranes to ships that have already berthed in order to perform container discharge/loading operations. It refers to both the "arrival of the ship" and "unloading/loading of the ship" processes. Its lifecycle is shown in Figure 14. A shuttle vehicle is a resource managed by the vehicle manager process. It refers to the "transport of containers from ship to stack and vice versa" process. Its lifecycle is shown in Figures 15 and 16.

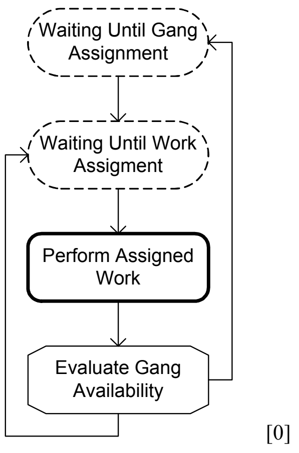

Waiting Until Gang Assignment
Waiting Until Work Assigment
Perform Assigned Work
Evaluate Gang Availability
[0]

Figure 15: HFC for Shuttle Vehicle Process

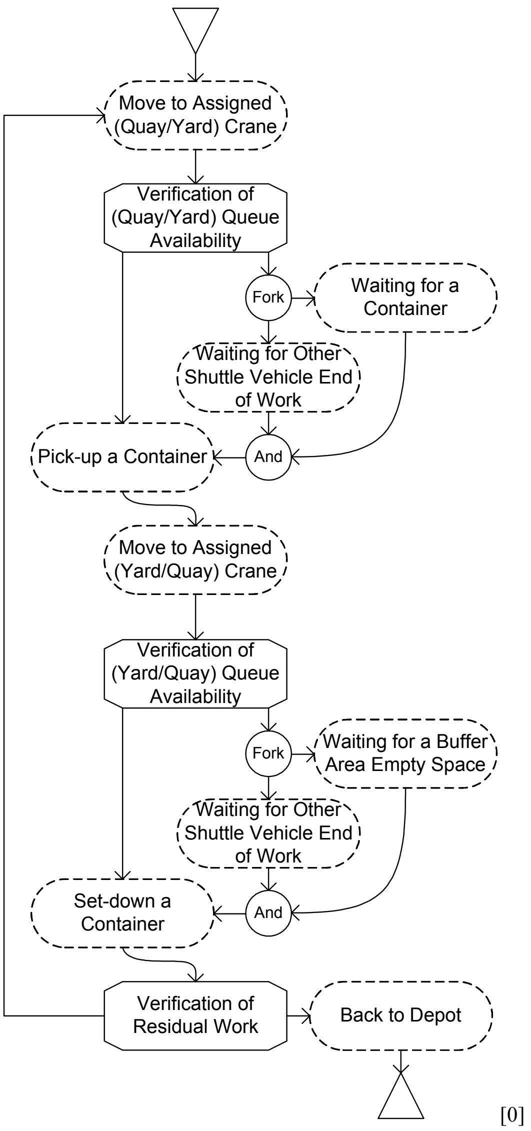

Move to Assigned (Quay/Yard) Crane
Verification of (Quay/Yard) Queue Availability
Fork
Waiting for a Container
Waiting for Other Shuttle Vehicle End of Work
Pick-up a Container
And
Move to Assigned (Yard/Quay) Crane
Verification of (Yard/Quay) Queue Availability
Fork
Waiting for a Buffer Area Empty Space
Waiting for Other Shuttle Vehicle End of Work
Set-down a Container
And
Verification of Residual Work
Back to Depot
[0]

Figure 16: HFC for Compound Activity "Perform Assigned Work" from the Shuttle Vehicle Process HFC

A carefull analysis of the processes illustrated through Figures 11, 12, 13, 14, 15 and 16 reveals the effectiveness of some methodological choices.

An example of model reusability is offered by Figures 11 and 13. In particular, the compound activity "Use of Port" for the ship process in Figure 11, reuses the ship process defined in Figure 13 within the berth planning model. "Use of Port" represents the set of operations related to the ship process that occur within the berth planning model.

The process described in Figure 12 is a typical example of the wholistic approach. In fact, the sub-model (berth planning) can be reused to allow the channel manager to control resource availabilty in the berth planning model in order to answer the port entrance request issued by an incoming ship.

The berth manager process, depicted in Figure 14, exhibits the major feature of being the indirect contact point between the berth planning model and a possible parallel model for the quay crane assignment problem (Canonaco et al. 2007). The compound activity "Crane Assignment" manages this connection according to the decisions provided by the operations manager process. In the Figures 15 and 16, a final example of reusability is shown. The compound activity "Perform Assigned Work" (Figure 15) is fully depicted in Figure 16. It does not have a unique representation and, in addition, it depends on the chosen yard management policy. So it is easily possible to customize the model in dependence of the system modelled.

## CONCLUSION

We have proposed a Modelling Paradigm to support the development of simulation models oriented to the optimal management of logistic activities in maritime container terminals. The MP uses a wholistic approach to capture the complex relationships among sub-systems and allows for a direct representation of the hierarchical structure of the decision making process for system management. A simulation quasi-language reveals its usefulness to fix the modelling choices once these are defined in compliance with the operations manager at a real terminal. System modelling is completed by a flow-chart based definition of processes involved in the model at hand. Real case examples of modelling have been presented, with the aim of supporting the future design of simulation based optimisation techniques. Currently, a Java tool is under development. To face the high burden on execution typical of PI world view simulation tool, we are also working on a parallel version of the simulator.

## REFERENCES

Canonaco, P.; P. Legato; R.M. Mazza; and R. Musmanno. 2007. "A Queuing Network Model for Management of Berth Crane Operations". Accepted on *Computers & Operations Research* – Special Issue – Queues in practice.

Cota, B.A. and R.G. Sargent. 1992. “A Modification of the Process Interaction World View”. *ACM Transactions on Modeling and Computer Simulation.* Vol 2, 2, 109-129.

Dahl, O. andf K. Nygaard. 1966. “SIMULA – an ALGOL-Based Simulation Language.” *Communications of the ACM*. 9 (9), 349-395.

Derrick, E.J.; O. Balci; and R.E. Nance. 1989. “A Comparison of Selected Conceptual Frameworks for Simulation Modeling”. In *Proceedings of the 1989 Winter Simulation Conference.* (Washington, DC, USA). 711-718.

Fritz, D.G. and R.G. Sargent. 1995. “An Overview of Hierarchical Control Flow Graph Models”. In *Proceedings of the 1995 Winter Simulation Conference.* (Arlington, VA, USA). 1347-1355.

Fu, M.C. 2001. “Simulation Optimization”. In *Proceedings of the 2001 Winter Simulation Conference*. (Arlington, VA, USA). Vol 1, 53-61.

Ghiani, G.; P. Legato; R. Musmanno and; F. Vocaturo. 2004. “Optimization via Simulation: Solution Concepts, Algorithms, Parallel Computing Strategies and Commercial Software”. *International Scientific Journal of Computing*. Vol 3, no 3.

Law, A.M. and W.D. Kelton. 2000. *Simulation Modelling and Analysis*, 3d ed. New York: McGraw-Hill.

Law, A.M. and M.G. McComas. 2002. “Simulation-based Optimization”. In *Proceedings of the 2002 Winter Simulation Conference.* (San Diego, CA, USA). Vol 1, 41-44.

Legato, P. and R.M. Mazza. 2001. “Berth Planning and Resources Optimisation at a Container Terminal via Discrete Event Simulation”. *European Journal of Operational Research*. 133, 537-547.

Moorthy, R. and CP. Teo. 2006. “Berth Management in Container Terminal: the Template Design Problem”. *OR Spectrum* (Special Issue in Container Logistics). Vol 28, 495-518.

Nance, R.E.. 1987. “The Conical Methodology: A Framework for Simulation Model Development”. In *Proceedings of the Conference on Methodology and Validation*. The Society for Computer Simulation. (San Diego, CA, USA). 38-43.

Nakayama, M.K. 2002. “Simulation Output Analysis”. In *Proceedings of the 2002 Winter Simulation Conference.* (San Diego, CA, Dec. 8-11). Vol 1, 23-34.

Notteboom, T.E. 2004. “Container Shipping and Ports: An Overview”. *Review of Network Economics*. Vol 3 (2), 86-106.

Ocean Shipping Consultants. 2003. *World Containerport Outlook to 2015*. OSC, Chertsey.

Overstreet, C.M. and R.E. Nance. 2004. “Characterizations and Relationships of World Views”. In *Proceedings of the 2004 Winter Simulation Conference*. (Washington, DC, USA). 279-287.

Pidd, M. and R.B. Castro. 1998. “Hierarchical Modular Modelling in Discrete Simulation”. In *Proceedings of the 1998 Winter Simulation Conference.* (Washington, DC, USA). Vol 1, 383-389.

Pidd, M. 2002. “Simulation Software and Model Reuse: A Polemic”. In *Proceedings of the 2002 Winter Simulation Conference.* (San Diego, CA, USA). Vol 1, 722-775.

Pidd, M. 2004. *Computer Simulation in Management Science*, 5th ed. Chichester: John Wiley & Sons.

Rizzoli, A.E.; L.M. Gambardella; M. Zaffalon; and M. Mastrolilli. 1999. “Simulation for the Evaluation of Optimised Operations Policies in a Container Terminal”. In *Proceedings of 1999 International WS on Harbour, Maritime & Logistics Modelling and Simulation.*

Sauer, C.H.; E.A. MacNair; and S. Salza. 1980. “A Language for Extended Queueing Network Models”. *IBM Journal of Research and Development*. Vol 4, no 6, 747-755.

Schulze, T.; M. Schumann; and G.D. Rehn. 2000. “Language Based Simulation Models as Management Tools for Assembly Lines”. In *Proceedings of the 2000 Winter Simulation Conference*. (Orlando, FL, USA). Vol 2, 1393-1401.

Sgouridis, S.P. and D.C. Angelides. 2002. “Simulation-Based Analysis of Handling Inbound Containers in a Terminal”. In *Proceedings of the 2002 Winter Simulation Conference.* (San Diego, CA, USA). Vol 2, 1716-1724.

Silver, G.A.; L.W. Lacy; and J.A. Miller. 2006. “Ontology Based Representations of Simulation Models Following the Process Interaction World View”. In *Proceedings of the 2006 Winter Simulation Conference*. (Monterey, CA, USA).

Steenken, D.; S. Voss; and R. Stahlbock. 2004. “Container Terminal Operation and Operations Research - a Classification and Literature Review”. *OR Spectrum* 26, 3-49.

Vis, I.F.A. and R. De Koster. 2003. “Transshipment of Containers at a Container Terminal: an Overview”. *European Journal of Operational Research*. 147(1), 1-16.

Zeigler, B.P. 1976. *Theory of Modelling and Simulation*. John Wiley and Sons, New York.

## AUTHOR BIOGRAPHIES

**PASQUALE LEGATO** is an Assistant Professor of Operations Research at Faculty of Engineering (University of Calabria), where he teaches courses on simulation for system performance evaluation. He has published on queuing network models for job shop and logistic systems, as well as on integer programming models. He has been involved in several national and international applied research projects and is serving as reviewer for some international journals. Current research activities are on development and analysis of queuing network models for logistic systems, discrete-event simulation and in the integration of simulation output analysis techniques with combinatorial optimization algorithms for real life applications in Transportation and Logistics. His home-page is `http://www.deis.unical.it/legato`.

**ROBERTO TRUNFIO** received the Laurea degree (*cum laude*) in Management Engineering from the University of Calabria, Rende, Italy, in 2005, and is currently pursuing the Ph.D. degree in Operations Research from the same university. Besides, he is logistics engineer at NEC Italy Center for High-Performance Computing and Computational Engineering (CESIC). His current research interests include modelling languages for discrete event simulation, simulation based optimisation of logistics systems, and parallel simulation.