EDUCATING DISCRETE SIMULATION BY AGENT-BASED ROLEPLAY

H.P.M. Veeke, J.A. Ottjes, G. Lodewijks
Dep. of Marine and Transport Technology
Faculty of Mechanical, Maritime and Materials Engineering, 3mE
Delft University of Technology
Mekelweg 2, 2628 CD Delft, the Netherlands
E-mail: H.P.M.Veeke@tudelft.nl, J.A.Ottjes@tudelft.nl, G.Lodewijks@tudelft.nl

Abstract

Since the introduction of real process oriented simulation it has been educated in two different ways. For students in informatics and mathematics it is educated in a strict formal way based on things like paradigms and finite state machines. For students that don't need to become professional programmers but do have to understand the principles of simulation it is mostly educated in an informal intuitive way when not learned by off-the-shelf click-and-play packages. Starting from a general programming platform, the major problem in educating simulation is the explanation of simultaneity and synchronization. This paper describes a recently developed method, by which students experience these problems themselves and –as far as the first results can show- master the techniques to solve synchronization problems. The method is based on roleplaying agents.

Keywords: simulation, process interaction, education, agent-based

1 Introduction

Many years of experience in educating simulation led us to the conclusion that the main problem is to let students understand how to describe unambiguously the "behavior of a system", in a time-based manner. Most discrete simulation literature [e.g. 1,2] starts from a notion of a change in the state of a system defining this notion as an "event". The 'event' however appears to be too abstract to understand completely and it leads to complex implementations of a system's behavior.

The real difficulty in understanding behavior is to realize that we implicitly take some "events" for granted, while they are essential for the synchronization of processing activities. "Doing nothing" is also a type of activity. This paper explains an agent-based role-play approach that improves the understanding and leads in a natural way to the process-oriented approach for describing the behavior of discrete systems.

2. Events vs. Processes

In [3] the construction of a "Tool for Object oriented Modelling And Simulation" (TOMAS) has been presented. It is implemented as a toolbox in the general programming platform Delphi®, so the complete functionality of Delphi® can be used too. Using Delphi® is not essential, but Delphi® is based on Pascal and this offers many advantages for students that are not supposed to become experienced programmers. Delphi® offers all possibilities and flexibility of a general programming language, so there will be no restrictions other than the creativity of the student or researcher. The way of modelling, closely matches the qualitative modelling as defined in the Delft Systems Approach (DSA) [4,5]. It differs widely from the approaches used in well-known packages. DSA uses as its main modelling element the concept of a "function" that expresses why a particular process is executed and what its contribution is to the environment. By this the modeller takes the necessary distance from what he experiences, in order to make a general model for the situation under investigation. Within this notion of function, a process is described from the company's viewpoint (as a repetitive series of activities of a department /group/person/machine that handles orders, materials, or even resources). Many packages use the viewpoint of the customer or the flowing element itself (a visitor's view). The visitor and company views are really different; for example, Zeigler et al. [6] call it the "flow oriented" vs. "real process oriented" approach. The latter approach is characterized by the fact that the sequence in which program statements are executed, differs from the written sequence.

An example is shown in table 1 for two elements A and B (the arrows show the order in which statements are executed). Already in the nineteen seventies a first implementation of the real process approach was constructed by Sierenberg and De Gans [7],called PROSIM. At that time the lectures about simulation with PROSIM didn't explain how to choose elements and corresponding processes. The lectures appealed to the common sense of the students, , which worked very well for some of the students, but left others in confusion.

ISBN: 1-60132-443-X, CSREA Press ©

Process of CompA	Process of CompB
Begin	Begin
CompB.Start(Now)	
Wait	
	Work(5)
	Color←Green
	CompA.Resume(Now)
While CompB.Color = Green	…
Standby	*Other things*
	…
	Color←Red
Finish	Finish
End	End

Table 1. Real Process Oriented Approach

The problem appeared to be twofold:

1. What is the selection criterion to choose elements?
2. How to describe and communicate the behaviour of the elements?

The first question has been answered in an earlier contribution [8], where it was found that recognizing the functions that need to be fulfilled, led to the elements fulfilling them. Here we will focus on the second question.

3. Behavior

A first short description will highlight the role of the elements in the model. Throughout this paper the example of an automated container terminal's import processes will be followed. Ships arrive from deep-sea at a berth of a container terminal. A number of containers should be unloaded, transported to a stacking area and stored there.

The (physical) elements in the model are Ships, Quay cranes, AGV's (Automated Guided Vehicles), ASC's (Automated Stacking Cranes), Stack and Containers.

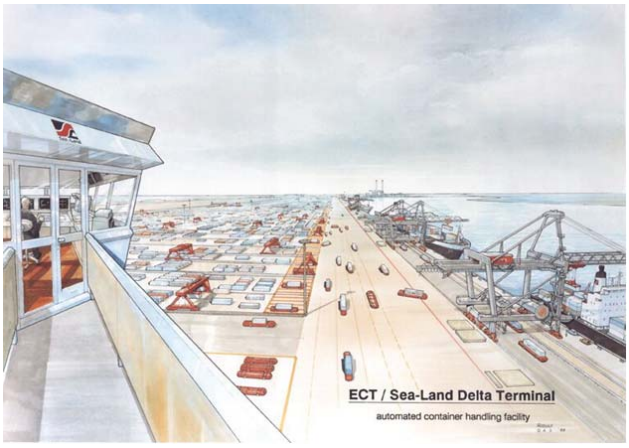

Fig.1. artist impression of Automated Container Terminal

The role of each element is:

Ship	: arrives with containers at berth
Quay crane	: unloads containers from ship
AGV	: transports containers from Quay crane to ASC
ASC	: stores containers into stack
Stack	: keeps containers in storage

Actually this is already a complete behavior description of the import processes, but there is no synchronization at all yet. Providing facilities to synchronize the different processes is the core problem of describing the behavior in order to construct a model like the one in the figure below.

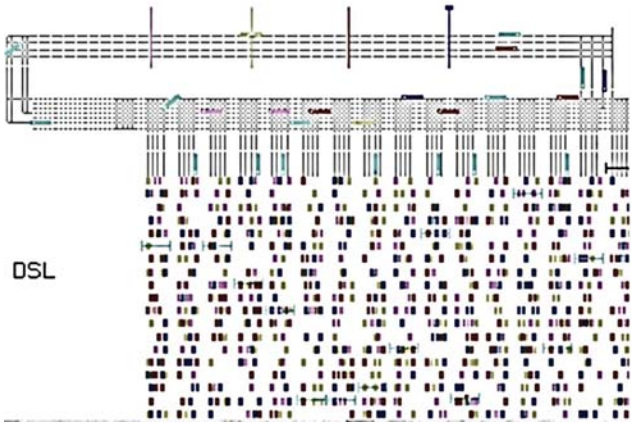

Fig.2. Screenshot of a model of container import processes [9]

We used to progress interactively with the students by expanding step by step the descriptions above. If we restrict the description to the synchronization between Quay cranes, AGV's and ASC's it could look like the table below.

Firstly it should be made clear that all equipment (or resources) repeat their actions during the whole simulation run, so its process description starts with "Repeat".

A Quay crane starts with unloading a container from a ship and needs a first synchronization with AGV activities; an AGV should be simply there to take over the container. Here we use a queue (QcQ) for this purpose, and let the Quay crane wait until there is at least one AGV in the queue. Queues are standard available in simulation packages, and one can use them for many purposes, here it is used for synchronization. Elements that have been placed in a queue should be removed from it too, so the quay crane removes the agv in front from the queue and puts a container on it. After that it signals the Agv by "Resume" to continue its independent part of the

process: driving to an ASC. At the ASC, the processes are synchronized in an analogous way.

Quay crane	Repeat … *Wait* While QcQ is empty Remove first AGV from QcQ *Put* Container on AGV Resume AGV …
AGV	Repeat Enter QcQ *Wait* *Drive* to ASC Enter ASCQ *Wait* *Drive* to Quay crane
ASC	Repeat … *Wait* While ASCQ is empty Remove first AGV from ASCQ *Lift* Container from AGV Resume AGV …

Table 2. Informal process descriptions

The "Process Description Language (PDL)" as presented in the table above, is being used to communicate on the behavior of the model. It is very useful both in teaching environments and practical design projects for the verification of the model. The big advantage of PDL is its simplicity and clarity, without the need of a special syntax and constructions that would be imposed by programming environments, I case we would try to describe the model's behavior immediately in some programming language or package.

We used to describe situations in this way interactively with the students, but noticed they had difficulties to reproduce it for other situations. The majority managed to define the elements correctly, but not all students were capable of reproducing this way of thinking on behavior in other situations. Many students stranded in an attempt to re-invent the basic provisions already available in the simulation toolbox. We apparently have to prevent that students consider the technical needs of a simulation environment as "modeling"; Instead they should focus on the synchronization needs of a modeling situation with the tools available.

We decided to adopt the agent based approach of programming and to replace each element with its corresponding agent, a straightforward conversion. An agent is the natural owner of a process and differs in nothing with a general simulation element.

4 Synchronization

Synchronization can be achieved in many different ways. First of all one could use a general type of semaphore that turns green or red to show "continue" or "stop". The disadvantage of this general approach is the loss of readability / understandability of the model.
We prefer to use the already available facilities of any simulation platform: attributes of elements and/or queues.
If the model contains an element "Fence" with a Boolean attribute Closed, one could easily make another element waiting for the Fence until it is open, by specifying "Wait while Fence.Closed = True" in its process description. It makes the use of semaphores very clear and natural in the descriptions. Even more powerful is the use of queues for synchronization. Many situations can be covered by one single queue status; it can be empty, it can be full, it can contain or just not contain one specific element. Depending on this status it is easy to stop or continue a process when one (or more) of these conditions is met.

We used to explain synchronization in this way, it seems trivial however when someone else is telling you how to implement it. Real difficulties arise when students have to construct it themselves.

In order to get the synchronization points clear, each student is assigned an agent of the model and together they should proceed in time as a system, a team of cooperating agents.

Each active time-consuming statement is assumed to take 5 seconds. In our example it concerns the statements Drive, and Put/Lift Container. Passive time-consuming statements should be solved with synchronization; it concerns the Wait statements in this case of which the time duration is unknown beforehand and depends on actions of other agents.

At the start of the role-play, each student is asked what his/her first action will be. The first question for each student should be: "Where am I?" Immediately followed by " What is my state?". Most of the students start mentioning actions, but forget these questions. They already assume implicitly that everybody knows where they are and in what state. This should be made explicit however, because it determines the starting point of the processes of other agents. It is also very important to decide on the starting state, because the behavior of most agents is repetitive and it should be easy to start the process in any way from this state. We assume an AGV is empty and starts waiting at the Quay crane by entering the QcQ.

When it is inside the QcQ, an AGV can only proceed after it has received a container from the quay crane. The Quay crane is the only one who determines this moment, so the only thing an AGV has to do is waiting; it doesn't have to stay looking (actively) until a container has been placed, it will get a signal from the Quay crane. The AGV agent waits until the Quay

crane agent tells it may continue. This is different from actively waiting like the Quay crane agent does. This agent waits until an AGV arrives in the QcQ, and after any "event" the agent should check this condition. It is very illustrative for the student to be asked by the lecturer every time to check this condition.

Now other students should start their process. The Quay crane will wait for a ship; after arrival it will start unloading container by container. When it has unloaded one container it will wait until an AGV arrives. If it is already there then it will put the container on the AGV, remove the AGV from AGVQ and wake up the waiting AGV by "Resume". The AGV-agent can proceed now. The lecturer may interrupt the agents of resources now and explain that Quay crane and AGV were synchronized by using a queue QcQ. Many times it happens that the agent Quay crane already proceeds without signaling the AGV etc.

5 Sequencing the processes of agents

Now both Quay crane and AGV proceed with their actions simultaneously (Quay crane unloading another container, AGV driving to ASC). The lecturer is keeping track of the time, and gives turn to the student that has a first activity. To make the picture complete one could pay attention to the fact that the teacher actually performs the role of "sequence mechanism".
The sequencing mechanism takes care of all state transitions and progress of time. During time an agent can be in one of three states:

1. Suspended or sleeping state. No moment in time has been defined for the agent to take action. It actually "sleeps" or plays the role of data element.
2. Scheduled state. A moment in time or a condition has been defined on which the agent should start or resume its actions.

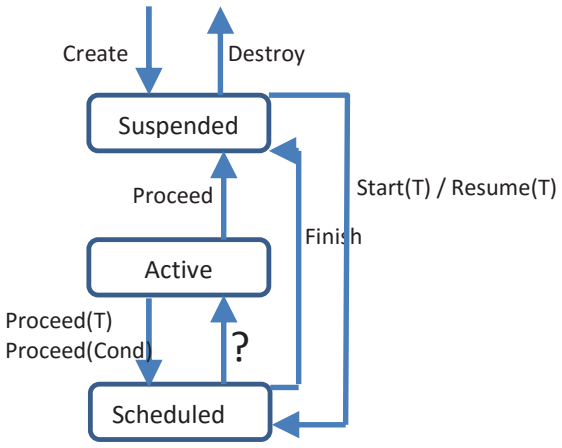

Fig. 3 Agent-states and transitions

3. Active state. The agent is actively executing actions (statements) until it tells the clock to proceed. The agent itself becomes scheduled or suspended then.

The question mark in the figure above shows that there is a mystery guest that changes states from "Scheduled" to "Active". This mystery guest is the core of the simulation toolbox that operates according the real process oriented approach.

The lecturer should explain which of the transitions is being used at any moment when another agent becomes active. He/she could also illustrate what happens if two agents become Active at exactly the same moment. It will then be immediately clear that there is no real simultaneity, because only one processor is available for executing the statements of the active agent; only one agent can be active at any time. Both ways of synchronization, attributes (e.g. fence closed) and queues (e.g. QcQ is empty) are sensitive for the order of activation by the simulation toolbox.

6 Conclusions and recommendations

In most cases simulation of logistic systems is explained from the viewpoint of an observer who has to construct the model. He is supposed to have the necessary knowledge of the system and describes the system in terms of events, activities and/or processes.
The real process-interaction approach is a method that can represent the real system and its dynamics in a very natural way. To teach students to apply this method we used agent based role playing in which students are asked to identify themselves with or, in other words, to step into the shoes of the various elements in the system and to live their lives (process) as a function of time. In this approach difficult issues like element interactions and synchronisation appear in a clear natural way. This leads to more insight of the operation of the real system and a more deeply understanding of the simulation model.

We recently started to use this approach in practice. A first course has been completed with this type of role-playing and the results seem to be promising. The students seem to understand the synchronization of simulation modeling better and used queues and attributes of agents in a natural way when developing their own models. Roleplaying works very explanatory.

Now we will apply this method also in our research projects with industrial partners in order to construct and verify simulation models of design situations, and clearly focus and decide on problematic synchronization cases.

Fig. 3 Agent-states and transitions
Fig. 3 Agent-states and transitions

7 References

[1] Fishman, G.S., "Discrete-Event Simulation", Springer-Verlag New York Inc.,, ISBN 0-387-95160-1, 2001
[2] Kleijnen, J.P.C., Groenendaal, W.J.H. van, "Simulatie: technieken en toepassingen",Academic Service, Schoonhoven, ISBN 90 6233 322 2, 1988
[3] Veeke, H.P.M., Ottjes, J.A., "TOMAS: Tool for Object-oriented Modelling And Simulation", Proc. Of Business and Industry Simulation Symposium, Washington, Ed. Maurice Ades, pp. 76 – 81, 2000
[4] in 't Veld, Prof. J., " Analysis of organisation problems", Wolters-Noordhoff bv, Groningen, 8th edition, ISBN 90-207-3065-7, 2002
[5] Veeke, H.P.M., Ottjes, J. A., Lodewijks, G., " The Delft Systems Approach", Springer,
ISBN 978-1-84800-176-3, 2008
[6] Zeigler, B.P., H. Praehofer, and T.G. Kim. 2000. "Theory of Modeling and Simulation", Academic Press, San Diego
[7] Sierenberg, R.W., de Gans, O.B., "PROSIM text book", lecture notes Delft University of technology, Delft, 1982
[8] Veeke, H.P.M., Ottjes, J.A., Lodewijks, G., "Experiences with process interaction based simulation in education and research" Proc. of the 2012 International Conference on Modeling, Simulation and Visualization Methods MSV 2012, Las Vegas, Ed. Hamid R. Arabnia, pp.109-113
[9] Duinkerken, M.B., Ottjes J.A., "A simulation model for automated container terminals", Proc. of the Business and Industry Simulation Symposium (ASTC 2000).
April 2000. Washington D.C. [SCS]. ISBN 1-56555-199-0