An Agent-based DES Supported Modeling Framework for Enterprises

Simona Iuliana Caramihai

POLITEHNICA University of Bucharest
Control Engineering and Computer Science Faculty
313 Splaiul Independentei, 77206 Bucharest, Romania
sic@ics.pub.ro

Abstract -The paper presents a unified modeling framework, agent based and supported by the Discrete Event System theory. This framework is intended to provide generic models, applicable at all enterprise levels, allowing evaluating interactions among different subsystems, in a complete customer-transparent manner. The evaluation is concerned with two main properties: meeting a given deadline for the whole process and allowing certain undesirable states as mutual blockings. The modeling procedure was explained for a simple case study of manufacturing cell. A software application for this evaluation procedure is under development; at this moment it allows to check for undesired states.

I. Introduction

The concept of interoperability was defined as "the ability for a system or a product to work with other systems or products without special effort of the part of the customer". More specific, interoperability could be seen as "the ability of Enterprise software and applications to interact […] The interoperability is considered to be achieved if the interaction can, at least, take place at the levels: data, application and business enterprise".[1]

From a system theory approach, this definition mainly results in primarily finding a common modeling framework for all the subsystems of a large scale heterogeneous system (the enterprise), then elaborating a methodology to combine modeling blocks according to given specifications and finally obtaining a possibility to compare different combinations from a certain performance point of view.

The common modeling framework is necessary for offering the same working environment for customers, without regard of systems/ products implied. The combining methodology is necessary for testing different possible interactions as multiple solutions to the same problem (which is the case for the majority of complex problems). Finally, the evaluation allows choosing the most appropriate solution for a given problem and a given set of objectives.

An appropriately chosen modeling support should allow stepwise extending models at different complexity levels.

From a generic point of view, the same kind of problem can be identified either as the real-time control of a (eventually flexible) manufacturing system or the control of a mobile (eventually intelligent) robot working into a low defined environment.

Similarities are: the complexity of the considered systems (a manufacturing system is composed of very numerous devices that should interact exchanging data and materials into a synchronized manner; a mobile robot is composed of an electro mechanic system moving according to information offered by sensors); their heterogeneity (devices composing a manufacturing system are from very different types, running different applications and changing different types of data and materials – as applications that should interact in enterprises; mobile robots have to synchronize mechanics, electronics and software for one single move and/or decision); the necessity to use simplified models – which leads to uncertainties, high dynamics of the processes, existence of multiple solutions for different problems, a.s.o.

Starting from these similarities, a possible solution for an interoperability-oriented modeling framework is to consider a modeling approach used for real-time manufacturing control.

The purpose of this paper is to present such an approach and to advocate it for interoperability.

2. Agent-based Modeling Approach

2.1. Context

One of the main characteristics of the interoperability approach as well as of the real-time manufacturing control, for instance, is that interactions between subsystems are not concerned with their detailed functioning.

Every subsystem can be seen as a black-box that can fulfill a certain list of non-decomposable activities (tasks). Those activities are interesting from the interaction point of view only as input/output data (for implementation purposes) and as having one of the following discrete states: *idle*, *in progress*, *break-down* (for state-space purposes).

The state of the overall system is obtained by composing the subsystems' states; the state evolution is obtained by the occurrence of events as *"start task"*/ *"end-task"*/ "*task breakdown*".

In manufacturing real-time control the goal is to obtain a certain desired state, by a sequence of events occurring from the initial one and, eventually to avoid other undesired states.

Basically, there are two evaluation criteria for manufacturing control architectures, that the majority design approaches proved as contradictory:

- Robustness (the capability of the control structure to bypass the effects of eventual resource/ activity break-downs in the controlled system so as to accomplish manufacturing goals – usually achieved by on-line rescheduling)
- Global optimality of the control policy in terms of system efficiency (one of the most important optimality criteria being the time spent by parts in the manufacturing system)

Consequently, there are two design approaches for control architectures: hierarchical and agent based.

In the case of hierarchical control architectures, the control problem is vertically stepwise decomposed at enterprise/ shop floor/ cell levels, where the basic criterion for differentiating levels is complexity (of data & process structure). Every level is

furthermore decomposed in modules, based on their respective functionality, thus resulting hierarchical distributed control structures.
These structures are usually very efficient from the point of view of the global optimality of the control policy, that can be off-line analyzed and evaluated. Other advantages are the clear modularization of the control system, implying the reusability of codes, lent degradation of performances in case of breakdowns, different frequencies of processing on different levels.
From the robustness point of view, performances of hierarchical distributed architectures are not very good: usually is necessary to prepare special procedures to treat breakdowns and it proves to be extremely complicated and costly, due to the complexity of the overall system.
The agent-based design approach is intended to ensure the robustness of the control system vs. breakdowns by the dynamic reconfiguration.
The basic idea is to use a functional horizontal decomposition of the control problem in order to create specialized modules, capable to locally solve, by dedicated algorithms, pre-defined goals. The main difference between these modules and those composing static hierarchical architectures is their position versus the environment. Without an hierarchical stratification, all the basic control functions - as task-planning, scheduling, monitoring, data upgrading and processing - need to be performed by the same module – the agent -, based on information obtained from other similar control entities.
Several heterarchical architectures [2], [3], [4] were developed for manufacturing control, having important advantages as the robustness to break downs and the facile extension of the control architecture with new modules but also important disadvantages of non-optimality and blocking.
One of the reasons of those disadvantages is that usually agents are not conceived to have a formal model and it is difficult to make off-line evaluation of solutions.
In the following a formal agent based model will be proposed in order to compensate the above mentioned disadvantages.

2.2. Proposed modeling framework

The modeling framework considers that every complex system could be interpreted as a network of servers and clients queuing for server provided services. Every service could have a given duration and a cost, thus allowing the evaluation of the overall system's performances.
Every client and every server are driven by *agents*, software structures capable of interactions and negotiations by a system of questions-answers. The approach is inspired by Lin and Solberg's negotiation model [2], by Albus' classical model of an intelligent agent [5] and is completed by a formal support.
The generic structure of an agent includes four basic blocks (Figure 1.):

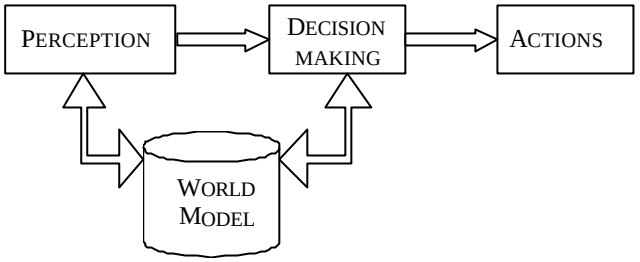


Fig. 1. Generic structure of an agent

- **The perception block** - receives every information from the environment and modifies, if necessary, the internal model of the agent;
- **The decision making block** – it decides either how to process the received data (with direct and explicit consequences on the immediate actions of the agent) or how to modify information and knowledge contained in the world model block (with implicit consequences on the whole behavior of the agent); usually based on the calculus of a multi-criterial cost function;
- **The action block** - is basically a communication interface with the environment: it either starts tasks performed by a server or sends typified questions for other agents;
- **The World Model block** - is a structured knowledge base containing the internal model of the agent (including relevant knowledge about the environment, i.e. the state of temporary links with other agents, its own) and a rule base on "how to take decisions".

The world model has to have intrinsic updating mechanisms for its two main parts, verifying and ensuring the global consistency of data.

2.3. Basic types of agents

There are two basic types of agents, every one having a specific internal organization. These basic types are the "client agent" and the "server agent".

The client agents are in charge with the realization of a specific product or of a list of activities, by negotiating with server agents controlling the necessary resources.

A server agent is controlling a specific resource in order to ensure processing for clients.

The communication between different types of agents is a negotiation procedure called “centered on the client” (i.e. client agents start negotiations and, if necessary, decide about the server agents they should choose).

The world model of a server agent should contain:

- the list of processing tasks that the resource could perform - with their respective duration and cost
- the (up-datable) current state of the resource
- the (up-datable) history of negotiations with client agents and their basic characteristics (solicited operations - cost requested - result of transactions)
- the (up-datable) efficiency (evaluation) function value

- information about agents with which are established in-course negotiations: solicited operations, cost, allocated priorities.

The decision making block should contain basically the rules for constructing an evaluation function for client-agents offers. These rules should be subject of modifications, based on the history of successful/ unsuccessful negotiations and of the current value of the efficiency function. Other rules included in the decision making part should be with regard to the realization of physical processing, allocation of auxiliary materials a.s.o.

As concerning a client agent, its world model could be more complex, including:

- for a product: all the technological available variants of the physical basic product; for an activity: the workflow of tasks composing the activity
- (up-datable) limits of cost functions allowed for different operations/ activities
- functional constraints as the due date, priority level, maximal costs, etc.
- the actual state
- the current value of its cost function and its history
- the upper time limit for duration
- the details of the in-course negotiations.

Negotiations should develop following a determined protocol and containing a typed set of questions-answers, both in order to simplify the communication procedure and mainly in order to finish in a given time limit.

At this level the communication protocol will be not detailed; it si considered as an implementation detail and should be transparent for the user.

In order to successfully finish a negotiation is necessary a mutual agreement of both parts, meaning that before allocating a server to a client, both agents should cancel other negotiations.

This condition implies that an agent could negotiate in the same time with several others, this approach being intended to optimize the efficiency of the system in terms of cost functions and considering that there could be several servers able to perform certain tasks.

In this case, imposing a maximum duration for a negotiation process of an agent could prove a good measure against the possibility to loose time trying to obtain the better server, especially when other alternatives are available.

3. Off-line evaluation and formal support

3.1. Evaluation and supervisory module

The proposed modeling approach could offer a support for considering into an unified way any kind of interaction into a system, from product-resource manufacturing level, to enterprise-enterprise into a virtual organization, provided that a communication protocol is ensured between negotiating agents. However, some difficulties can arise when there are concerned several servers and several dient competing for the same resources, as for instance:

- a critical server is requested by two clients with comparable cost functions for the operations in cause - blocking situation
- two (or more) clients are needing at least two servers each for performing their tasks; every one has reserved one server and needs the servers reserved by others (the Chinese philosophers problem)

- a client agent lasts in the system more than its upper time limit because its negotiating procedure is not enough "aggressive" compared with other agents – delay situation

For solving and/ or preventing this kind of problems and for ensuring a global degree of optimality, an off-line performance evaluation of negotiations could be performed. The software module that can perform such an evaluation will be denoted as the *supervisory* module.

A supervisory module can evaluate a given group of agents. Considering their respective internal models – goals included – the supervisor is intended to combine them into a global model and to off-line synthesize and evaluate all their possible interactions. Finally it can select and recommend the optimal one.

The consistency of the evaluation is ensured by the modeling support of agents/ world models.

3.2. The modeling support

The structure of agents is designed so as the following aspects are distinct:

- the environment - represented dually by its model and its perception as feedback for actions
- the goals - decomposable in tasks
- the methods for executing tasks and for decomposing goals
- the actions on the environment as the results of tasks

Based on these considerations it results that the world model would have a dual representation: in terms of rules for goals, methods and actions and in a dedicated formal support for the environment.

This modeling support should meet some minimal conditions as:

- to reflect the typical behavior of client/ servers
- to permit modeling at different complexity levels agents and their goals
- to allow a modular approach
- to integrate time,
- to allow condition/ rules driven evolution of models but also quantitative evaluation

A formalism meeting these requirements are (timed) Petri Nets [6]. The off-line evaluation by a supervisor is based on the principles presented in [7].

An off-line evaluation procedure will have the following steps:

1. Selection of the interesting agent models - from the client agents world model (implying decisions if the clients have different variants of action)
2. Selection of the necessary server models - taking into account only the operations requested by the actual clients
3. Construction of the closed loop functioning model, based on the synchronous composition of models
4. Evaluation

Case study:

There is a system consists of two servers, M_1 and M_2, each of them being capable of performing two types of operations ($op_i\ M_j$ – i is the operation number, j is the server that makes the operation):

$M_1 : op_1\, M_1, op_2\, M_1$; $M_2 : op_1\, M_2, op_2\, M_2$

Two products, Pr_1 and Pr_2, are to be realised in parallel, each of them requiring two operations,

$Pr_1 : op_1 M_1, op_1 M_2$; $Pr_2 : op_2 M_1, op_2 M_2$

The system could be modeled by two server agents representing M_1 and M_2, and two client agents, Pr_1 and Pr_2. The sequences of operations for the client agents represent a restriction for the system evolution.

The PN model of the global system is presented in Figure 2. In Table 1 are presented the signification of the places (P_i) and of the transitions (T_j). Usually, in PN models transitions model events and/or actions and places – states and/or conditions. A marking in a place means that the respective condition is fulfilled.

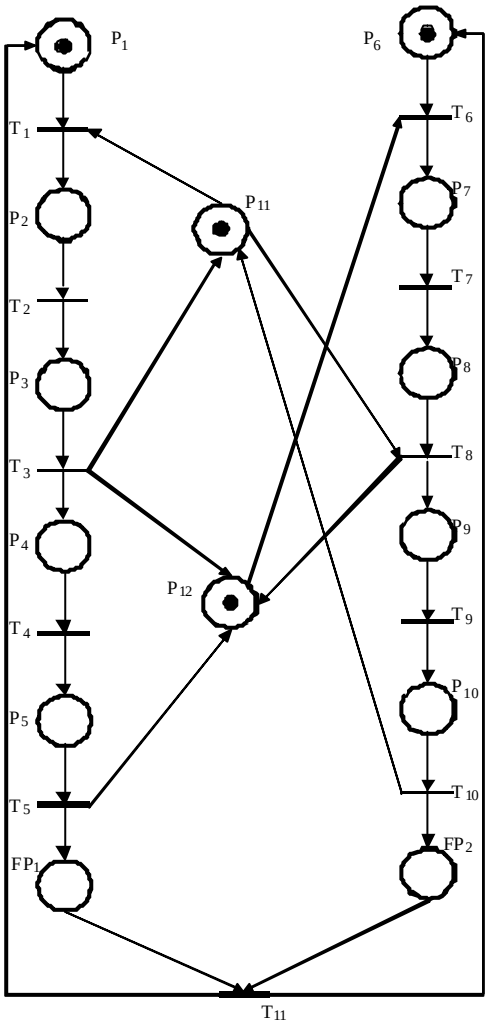


Fig. 2. : PN model of the FMS

Table 1. Node significance

P_1	Number of parts of Pr_1 type
P_2	Processing a part Pr_1 on the resource M_1
P_3	Parts Pr_1 processed on M_1
P_4	Processing a part Pr_1 on the resource M_2
P_5	Parts Pr_1 processed on M_2
P_6	Number of parts of Pr_2 type
P_7	Processing of a Pr_2 part on the resource M_1
P_8	Parts Pr_2 processed on M_2
P_9	Processing of a Pr_2 part on the resource M_2
P_{10}	Parts Pr_2 processed on M_1
P_{11}	M_1 free
P_{12}	M_2 free
FP_1	Pr_1 ended the operations
FP_2	Pr_2 ended the operations
T1	*Begin of processing Pr1 on the M1*
T2	*Uncontrollable End of processing Pr1 on the M1*
T3	*Begin of processing Pr1 on the M2*
T4	*UncontrollableEnd of processing Pr1 on the M2*
T5	*End evolution of Pr1*
T6	*Begin of processing Pr2 on the M2*
T7	*UncontrollableEnd of processing Pr2 on the M2*
T8	*Begin of processing Pr2 on the M1*
T9	*UncontrollableEnd of processing Pr2 on the M1*
T10	*End evolution of Pr2*
T11	*End of the cycle of the manufacturing*

When a client agent desires to perform a given operation, it simply requests for servers able to perform it. The server agents whose resources are idle and able to perform the operation answer to the request, giving additional information on the duration and quality parameters the resource can offer. A negotiation starts between the client agent and the servers whose information is convenient, aiming to establish a mutually acceptable cost of the operation.

All the possible interactions of the agents could be obtained by analyzing the state space of the Petri Net.

By specifying undesired states – as blockings – the supervisory module can eliminate those interactions leading to them. In [7] is presented in detail a methodology for the elimination of interactions leading to undesired states. A software application was already designed for implementing this methodology for untimed systems. Figure 3 is presenting the state space of solutions for the case study, as well as the acceptable state space, obtained after eliminating blockings.

Petri Nets allows for a very easy inclusion of time in the logical model – by simply associating durations to the nodes modeling activities. Timed model allow for comparing solutions from the duration point of view and – by extension – by every defined cost function.

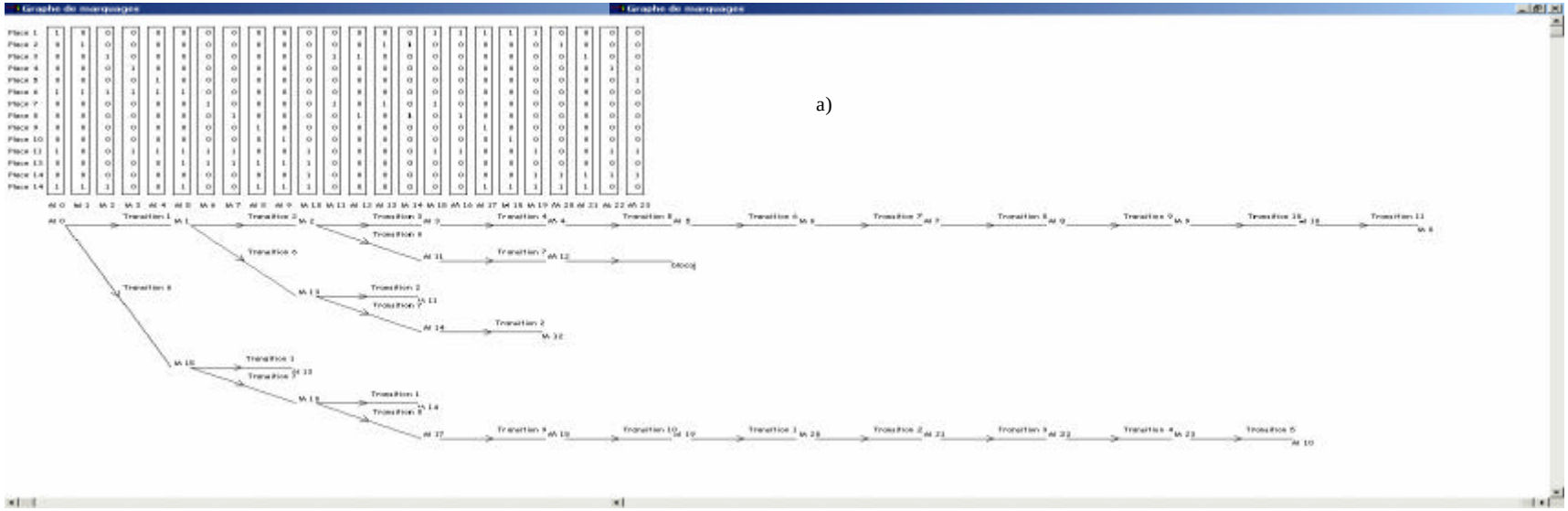
Conclusion

The paper presents a modeling framework that allows evaluating interactions between heterogeneous subsystems of a complex structure. The main hypothesis is that every complex system could be modeled as a client/server queuing network and consequently represented as a two-type agent based structure. The evaluation of agent interactions and the elimination of possible problems as blocking could be made based on the synchronous composition of the Petri Nets world models of the agents. The agent composition as well as the evaluation is supported by a dedicated software application.

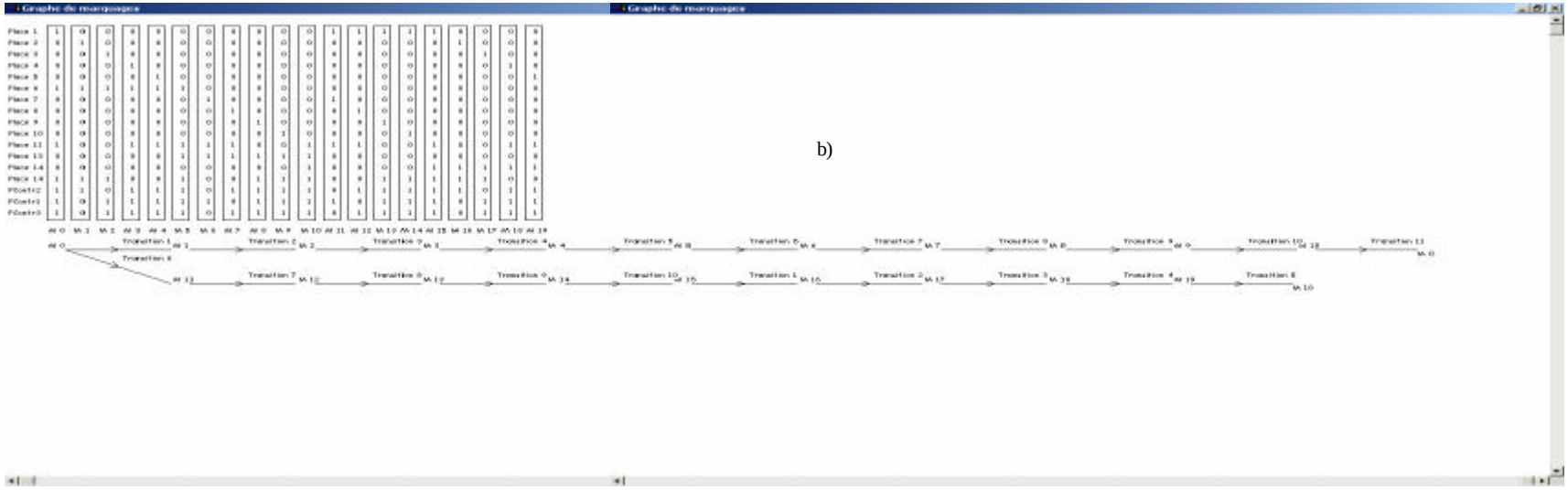
Future research directions are the evaluation of timed models and the realization of a software shell for creating an agent library.

References

1. http://interop-noe.org/INTEROP/presentation, page.2
2. Lin, Y., Solberg, J. "Autonomous control for open manufacturing systems", Computer Control of Flexible Manufacturing Systems. (Joshi S. and Smith J (Ed.)), Chapman & Hall, 1994
3. McFarlane, D. "Holonic manufacturing systems in continuous processing: concept and control requirements", Proceedings of the Annual Conference of ICIMS-NOE, ASI '95, pp. 273-282, Cascais, Portugal, June 25-28, 1995
4. Wyns, J., Van Brussel, H., Valckenaers, P. "Design pattern for deadlock handling in holonic manufacturing systems", Production Planning and Control, Vol. 10, No. 7, pp. 616-626. 1999
5. Albus J.S., Barbara A.J., Nagel R.N., "Theory and practice of hierarchical control", Proc. IEEE Computer Soc.Int.Conf Compcon Falls, National Bureau of Standards, Washington DC, 1995
6. Rene David, Hassane Alla: Discrete, Continuous and Hybrid Petri Nets, Springer, 2004
7. CIFA sau similar (CSCS ?)



a)



b)

Fig. 3: The state space:
a) complete b) desirable