Exploiting the easyABMS methodology in social and economic domains

Alfredo Garro
Dipartimento di Elettronica,
Informatica e Sistemistica (D.E.I.S.)
Università della Calabria
Via P. Bucci 42 C
87036, Rende (CS), Italy
alfredo.garro@unical.it

Wilma Russo
Dipartimento di Elettronica,
Informatica e Sistemistica (D.E.I.S.)
Università della Calabria
Via P. Bucci 42 C
87036, Rende (CS), Italy
w.russo@unical.it

ABSTRACT

Agent-Based Modeling and Simulation (ABMS) represents a powerful approach for analyzing and modeling modern social and economic systems as they can be naturally conceived as composed of autonomous, goal-driven and interacting entities (agents) organized into societies. However, although several tools for ABMS are available, there are few methodologies and related processes which are able to cover all the phases from the analysis of the system under consideration to its modeling and simulation results analysis. Moreover, the absence of visual modeling tools and techniques for ABMS often constitutes an entry barrier for whoever lacks advanced programming skills. This paper presents and exemplifies through a case study an integrated and iterative methodology (easyABMS) specifically conceived for agent-based modeling and simulation of complex systems which is able to support domain experts in fully exploiting the benefits of the ABMS while significantly reducing programming and implementation efforts. The case study, concerning the management of a three-stage supply chain, shows how easyABMS can be effectively exploited for the agent-based modeling and simulation of modern social and economic systems.

Keywords

Agent-Based Modeling and Simulation, Supply Chain Management, Agent-Oriented Methodologies.

1. INTRODUCTION

For solving a wide range of social and economic problems, several models of social and economic systems were developed by adopting in most case (strong) simplifying assumptions which make these models representative of the modeled system only in particular and restrictive conditions. The simplicity of the developed models was due not only to the nature of the problems to be solved but also by the available tools for building and managing these models. As the (social and economic) problems to face are becoming more challenging and the descriptive and predictive capabilities of the related models are becoming, likewise, more challenging too, the traditional simplifying assumptions and their related traditional tools are demonstrating inadequate, and the exploitation of new and more powerful tools for representing and managing a *level of complexity*, which is more adequate for satisfying these new and challenging purposes, is more and more required. In this context, Agent Based Modeling and Simulation (ABMS) represents a new and powerful way for analyzing and modeling complex systems as it is able to fully represent a system at different levels of complexity in terms of autonomous, goal-driven and interacting entities (agents) organized into societies which exhibit emergent properties, that is, properties which arise from the interactions between the component entities and that cannot be deduced *a priori* simply considering only the properties of the individual entities. The agent-based model of a system is, then, executed to simulate the behavior of the complete system so that knowledge of the behaviors of the single entities (micro-level) can produce an understanding of the overall outcome at the system-level (macro-level).

To date, although several tools for ABMS are available [9, 10, 16, 17, 21] as well as methodologies for the development of agent-based systems which are mainly proposed in the context of Agent-Oriented Software Engineering (AOSE) [5], there are only a few methodologies and related processes which are able to seamlessly guide domain experts with limited programming expertise from the analysis of the system under consideration to its modeling and subsequent simulation results analysis [6, 7, 14].

To address these issues, this paper presents easyABMS [3,4], a methodology specifically conceived for agent-based modeling and simulation of complex systems, and exemplifies its effectiveness in the social and economic domains through a case study which concerns the management of a three-stage supply chain. easyABMS aims at supporting domain experts in fully exploiting the benefits of the ABMS while significantly reducing programming and implementation efforts; in particular, easyABMS defines a process which is: (i) *complete* as its phases cover from the analysis of the system under consideration to its modeling and simulation analysis; (ii) *integrated* as each phase refines the model of the system which has been produced in the preceding phase; (iii) *visual* as the work-products of each phase are basically different models of the system mainly constituted by visual diagrams based on the UML notation [20]; (iv) *model-driven* as according to the Model Driven paradigm [1, 18] the simulation code is automatically generated from the obtained Simulation Model of the system; (v) *iterative* as, on the basis of the simulation results, a new/modified and/or refined model of the system can be obtained through a new process iteration which can involve all or some process phases.
The remainder of this paper is organized as follows: Section 2 presents an overview of the easyABMS methodology and the related process; Section 3 shows its application to the agent-based

modeling and simulation of a three-stage supply chain; finally, conclusions are drawn and future works delineated.

Table 1. easyABMS: process phases, work products and main related concepts.

Process Phase	**System Analysis**	**Conceptual System Modeling**	**Simulation Design**	**Simulation Code Generation**	**Simulation Set-up**	**Simulation Execution**	**Simulation Results Analysis**
Work Product	Analysis Statement	Conceptual System Model : ▪ Structural System Model ▪ Society Model ▪ Agent Model : o Goal Model o Behavioral Model o Interaction Model ▪ Artifact Model : o Behavioral Model o Interaction Model	Simulation Model : ▪ Simulation Context Model ▪ Simulation Agent Model	Simulation Code	Simulation Scenarios	Simulation Results	Simulation Analysis Reports
Main Concepts	Composed entity	Society	Simulation Context	Java Classes	Depending on the features of the exploited Simulation Framework		
	Pro-active entity	Agent	Simulation Agent				
	Re-active entity	Artifact					
	Passive entity	Artifact (Resource Manager of the passive entity)					
	intra-entity relationship	Interaction	Interaction Link among Simulation Agents				
	inter-entity relationship						

2. easyABMS : AN INTEGRATED METHODOLOGY FOR ABMS

The easyABMS methodology defines a process for ABMS composed of seven subsequent phases from the preliminary *System Analysis* to the *Simulation Result Analysis*. On the basis of the obtained simulation results a new iteration of the process which can involve all or some process phases can be executed for achieving new or not yet reached simulation objectives. Specifically, the process phases are the following:

– *System Analysis*, in which a preliminary understanding of the system and the main simulation objectives are obtained (Analysis Statement);
– *Conceptual System Modeling*, in which a model of the system is defined in terms of agents, artifacts and societies (Conceptual System Model);
– *Simulation Design*, in which a model of the system is defined in terms of the abstractions offered by the framework which is exploited for the simulation (Simulation Model);
– *Simulation Code Generation*, in which the Simulation Code for the target simulation environment is automatically generated starting from the model which is obtained in the previous phase;
– *Simulation Set-up*, in which the Simulation Scenarios are set;
– *Simulation Execution* and *Results Analysis*, in which the simulation results are analyzed with reference to the objectives of the simulation identified in the *System Analysis* phase.

The phases related to simulation exploit the Repast Simphony Toolkit [15, 17], which is the most popular ABMS toolkit and provides advanced features of visual modeling of agent behaviors and (semi)automatic code generation; moreover, for the phase of *Simulation Results Analysis*, the Toolkit supports an integrated use of several powerful analysis tools (Matlab, R, VisAd, iReport, Jung).

For each process phase the work-products and main related concepts are reported in Table 1 whereas a briefly description is given in the following sub-sections; a more complete description can be found in [3,4].

2.1 System Analysis

In the System Analysis phase the user specifies the objectives of the simulation and analyses the system being simulated so to obtain a preliminary understanding of the system and its organization.

The *System Analysis* phase, which is based on the principle of *layering* and exploits the well-known techniques of *Decomposition*, *Abstraction* and *Organization* [2,8], is constituted by a sequence of *analysis steps*. In each *step* the user produces a new system representation by applying the *in-out zooming mechanisms* [11] to the entities which compose the system representation resulting from the preceding *analysis step*. The entities which are not *zoomed* among two consecutive *steps* are said to be *projected*. As the system is itself a (composed) entity, in the first *analysis step* the user chooses the starting level of abstraction for analyzing the system and *zooms-in* on it.

An entity can be characterized by an autonomous and goal-oriented behavior (*pro-active entity*), by a pure stimulus-response behavior (*re-active entity*), or can be *passive*; moreover, both the rules governing entities and their evolution, and the relationships among entities are specified. Specifically, *Safety* rules determine the *acceptable and representative* states of an entity whereas *liveness* rules determine which state transitions are feasible during the entity evolution. Relationships can be either *intra-entity relationships* (i.e. relationships among the component entities obtained by the zooming-in of an entity) or *inter-entity relationships*.

The *System Analysis* phase ends when the user obtains a *System Representation* of the system in which each component (pro-active, re-active, passive) entity has been represented at the level of abstraction which is appropriate for the objectives of the simulation. This *System Representation* along with a synthetic description of the system being considered, a detailed description of each identified entity, and the objectives of the simulation constitutes the work-product of this phase (the ***Analysis Statement***).

2.2 Conceptual System Modeling

The starting point of the *Conceptual System Modeling* phase is the *System Representation* resulting from the *System Analysis* phase in terms of atomic/composed entities, their relationships, and their rules.

Main concepts of this phase and the derivation rules from the concepts of the *Analysis Phase* are reported in Table 1; the exploitation of these rules straightforwardly leads to the first work-product of the phase (the ***Structural System Model***).

For each entity in the *Structural System Model* is then defined a specific model which depends on the entity type (*Society Model*, *Agent Model*, *Artifact Model*).

In particular:

- a ***Society Model*** details the entities which compose a *Society*, their type (*Agent*, *Artifact*, *Society*), and the rules governing the *Society* (*safety* rules) and its evolution (*liveness* rules);
- an ***Agent Model*** details the complex goal of an *Agent* (***Agent Goal Model***), its behavior (***Agent Behavioral Model***), and its interactions with other *Agents* and *Artifacts* in which the agent is involved (***Agent Interaction Model***);
- an ***Artifact Model*** details the behavior of an *Artifact* (***Artifact Behavioral Model***), and its interactions with other *Artifacts* and *Agents* (***Artifact Interaction Model***).

2.3 Simulation Design

Given the *Conceptual Model* of the system, in this phase the user obtains a model of the system in terms of the abstractions offered by the framework exploited for the simulation. Currently, easyABMS adopts as reference simulation framework the *Repast Simphony Toolkit* [15, 17]. Specifically, the *Simulation Design* and the *Simulation Code Generation* phases are supported by the *Repast Simphony Development Environment* [12], whereas the *Simulation Set-up*, the *Simulation Execution* and the *Simulation Results Analysis* phases are supported by the *Repast Simphony Runtime Environment* [13]. The Repast Simphony Toolkit was chosen as the most popular ABMS toolkit [14] and provides both the advanced features of visual modeling of agent behaviors and the (semi)automatic generation of code [15]. Moreover, several powerful analysis tools as Matlab, R, VisAd, iReport, Jung, can be directly invoked from the *Repast Simphony Runtime Environment* [13].

The *Simulation Model* is obtained by exploiting the derivation rules as they emerge from the relationships among the main concepts of each phases reported in Table 1. Specifically, each *Society* becomes a Repast Simulation Context (*SContext*): the System is the root *SContext* and any enclosed *Society* is a (sub)-Context of the corresponding enclosing *Society*. As *Artifacts* and *Agents* become Repast Simulation Agents (*SAgents*), the *Activities* which compose their behaviors are easily converted into Repast Simulation Behaviors (*SBehaviors*); moreover, the relationships derived from *Interactions* among *Agents* and *Artifacts* generate *Repast Network Projections*.

2.4 The other Simulation related phases

According to the Model Driven paradigm [1, 18], the *Repast Simphony Development Environment* [12] is able to automatically generate a great part of the simulation code from the obtained *Simulation Model* of the system. The user can therefore access and modify all the generated code extending it with additional Java and XML code. The obtained code is compiled by the *Repast Simphony Development Environment* using a Java compiler and then loaded in the *Repast Simphony Runtime Environment*.

Before starting the simulation the user *sets*: (i) the simulation scenario by specifying the values of the simulation parameters defined in the Simulation Design phase; (ii) the presentation preferences for the simulation results concerning the system properties of interest identified during the Simulation Design phase.

The simulation of the system is then executed by the *Repast Symphony Runtime Environment* on the basis of the specified simulation parameters. The simulation results concerning the system properties of interest for the user are presented to the user on the basis of the choices made during the simulation set-up.

Finally, the user analyses the simulation results, also by exploiting the analysis tools (Matlab, R, VisAd, iReport, Jung) which can be directly invoked from the *Repast Simphony Runtime Environment* so to verify whether the objectives of the simulation individuated during the *System Analysis* phase have been achieved. Where objectives have not been achieved or where new simulation objectives emerge, the user can execute a new iteration of the process which can then involve all or some process phases so that the new/modified and/or refined models of the system make it possible to achieve the remaining/new simulation objectives.

3. AGENT-BASED MODELING AND SIMULATION OF A THREE-STAGE SUPPLY CHAIN

To show how the easyABMS methodology can be effectively exploited for the agent-based modeling and simulation of modern social and economic systems a supply chain management scenario is considered. Specifically, the reference scenario, inspired by the well-known *beer game* [19], concerns a supply chain constituted of production companies (*producers*), carrier companies (*carriers*), and sales companies (*vendors*). A *producer* produces a single type of perishable good, manages orders received by *vendors* and uses a *carrier* for delivering the ordered quantity of goods to the ordering *vendor*. A *vendor* sells goods to final consumers and manages its own stock of goods.

3.1 System Analysis

In this phase a *System representation*, which highlights its component entities (pro-active, re-active, passive) and their relationships, is obtained. In particular, the level of abstraction of each component entity, which is obtained by applying the *in-out zooming mechanisms* during the different analysis steps, strongly depends on the objectives of the simulation (see Section 2.1). With reference to the supply chain under consideration, an agent-based model could be defined and simulated in order to compare and evaluate different *production*, *pricing* and *stock management* policies which *producers* and *vendors* wish to adopt to maximize their respective profits by maximizing incomes and minimizing costs [19]. In particular, a *producer* may periodically decide on the amount of goods to produce and the corresponding price, and a *vendor* may periodically establish the price of the good and the amount of goods to order. In this context, the simulation aimed to compare three different *production a*nd *pricing* policies for a producer to obtain both qualitative and quantitative information about them and their main parameters [7, 19]:

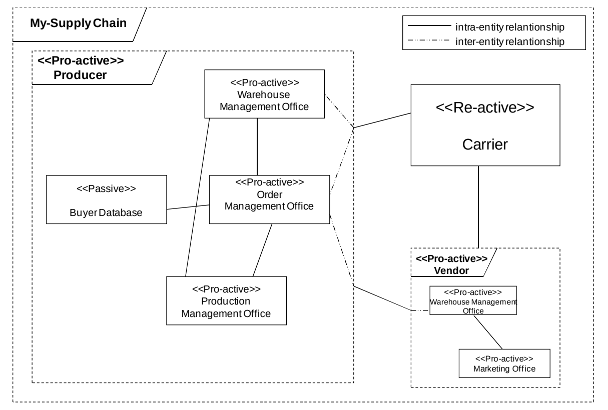
- *changeless*: the monthly production and the product price which have been fixed during the simulation set-up never change during the simulation execution;
- *incremental*: if the last month revenue has increased from the previous month, the monthly production and the product price increase by ΔPr and ΔPp respectively, otherwise their values are those of the last month;
- *adaptive*: if the last month revenue has increased from the previous month, the monthly production and the product price increase by ΔPrI and ΔPpI respectively, otherwise the monthly production and the product price decrease by ΔPrD and ΔPpD respectively.

The *System Representation* obtained on the basis of the identified simulation objectives is reported in Figure 1a. It is worth noting that the *Producer* and the *Vendor* entities have been *zoomed-in* during the analysis steps (see Section 2.1). These identified entities are further described, along with their relationships and their *safety* and *liveness* rules, in a textual format enriched by tables and diagrams which are not reported due to space limitations.

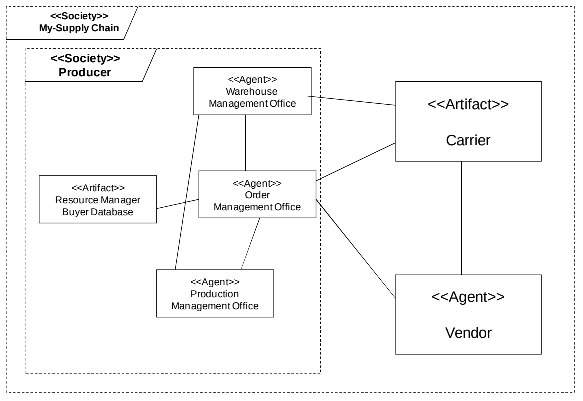
3.2 Conceptual System Modeling

The *Structural System Model* derived from the *System Representation* which is obtained from the *System Analysis* phase (Figure 1.a) is reported in Figure 1.b; in particular, as the focus is on the *Producer*, in the first iteration of the process, the level of representation chosen for the *Vendor* is more abstract with respect to the level resulting from the Analysis phase and the relationships refer to the involved *Agents* and/or *Artifacts*, thus crossing the boundaries of the *Societies*.

For each entity in the *Structural System Model* is defined the corresponding *Society*, *Agent or Artifact Model*s (see Section 2.2). Due to limitations space, in the following sub-section only the *Society Model* for the *Producer* Society, the *Agent Model* for the *Vendor* Agent and the *Artifact Model* for the *Carrier* Artifact are reported.


My-Supply Chain
intra-entity relantionship
inter-entity relantionship
<<Pro-active>> Producer
<<Pro-active>> Warehouse Management Office
<<Passive>> Buyer Database
<<Pro-active>> Order Management Office
<<Pro-active>> Production Management Office
<<Re-active>> Carrier
<<Pro-active>> Vendor
<<Pro-active>> Warehouse Management Office
<<Pro-active>> Marketing Office

(a) An overview of the System Representation obtained in the System Analysis phase


<<Society>> My-Supply Chain
<<Society>> Producer
<<Agent>> Warehouse Management Office
<<Artifact>> Resource Manager Buyer Database
<<Agent>> Order Management Office
<<Agent>> Production Management Office
<<Artifact>> Carrier
<<Agent>> Vendor

(b) An overview of the Structural System Model obtained in the Conceptual Modeling phase

Figure 1. System representations resulting from the Analysis and the Conceptual Modeling phases.

3.2.1 The Producer Society Model

The *Society Model* of the *Producer Society* (see Figure 1.b) is shown in Figure 2 in which the different entities which compose the *Producer Society*, and the *safety* and *liveness* rules governing the *Society* and its dynamics are reported.

Entity	Type
Order Management Office	Agent
Production Management Office	Agent
Warehouse Management Office	Agent
Buyer Database	Artifact (Resourc e Manager)

Safety rules

S_Prod1. WS (t) = PG (t) – SG(t) – DG(t);

where WS(t) is the warehouse stocks at time t; PG(t) is the quantity of goods that have been produced until time t; SG(t) is the quantity of goods which have been used to fulfill orders until time t; DG(t) is the quantity of goods which, at time t, have been eliminated due to expiration.

S_Prod2. ...

Liveness rules

L_Prod1. The Order Management Office cannot start satisfying an order that has not been correctly and completely received.

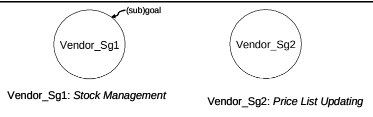
L_Prod2. …

Figure 2. A part of the Society Model of the Producer Society.

3.2.2 *The Vendor Agent Model*

Part of the *Agent Model* of the *Vendor* Agent is shown in Figure 3. In particular:

- Figure 3.a shows the *Vendor Goal Model* where the two goals (*Stock Management* and *Price List Updating*) which compose the complex goal of the *Vendor Agent* are specified along with their achievement relationships (in this case the two goals can be achieved independently);
- Figures 3.b illustrates a part of the *Vendor Behavioral Model*; in particular, the *Vendor Activity Table* which specifies the activities (*Order Planning, Goods Reception,* and *Price Definition*) which the *Vendor* Agent executes for achieving its goals, along with the pre and post conditions and the execution schedule (periodical or triggered). Each activity in the Agent *Activity Table* is further described by: (i) an UML [20] *Activity Diagram* which details the flow of execution (*control flow*) of the actions into which the activity can be decomposed; (ii) an *Activity Action Table* which reports, for each single action, a synthetic description of the action along with its pre and post conditions, the capabilities required for carrying out the action and its type (*computation or interaction*). As an example, the figure shows the UML *Activity Diagram* for the *Order Planning* activity;
- Figure 3.c reports the *Vendor Interaction Model* which specifies, for each action of the *interaction* type, the activity in the *Agent Activity Table* in which the interaction appears along with the *initiator*, the *partners* of the interaction, and the *exchanged information*.

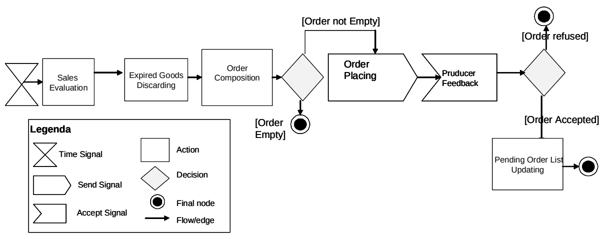

(sub)goal
Vendor_Sg1
Vendor_Sg2
Vendor_Sg1: *Stock Management*
Vendor_Sg2: *Price List Updating*

(a) *The* ***Vendor Goal Model***

- Vendor Activity Table -

Activity	**Goal**	**Pre-cond.**	**Post-cond.**	**Execution schedule**
Order Planning	Vendor_Sg1	…	…	Periodical
Goods Reception	Vendor_Sg1	…	…	Triggered
Price Definition	Vendor_Sg2	…	…	Triggered

- UML Activity Diagram for the Order Planning Activity -



(b) *A part of the* ***Vendor Behavioral Model***

Interaction	**Activity**	**Initiator**	**Partners**	**Exchanged Information**
Order Placing	Order Planning	Vendor	Order Management Office	Order Data
Producer Feedback	Order Planning	Order Management Office	Vendor	Order Data Decision on order acceptance
Goods Delivering	Goods Reception	Carrier	Vendor	Delivery Date
New Price	Price Definition	Order Management Office	Vendor	Product Price

(c) *The* ***Vendor Interaction Model***

Figure 3. A part of the Agent Model of the Vendor Agent.

3.2.3 *The Carrier Artifact Model*

As for the *Agent Model* (see Section 3.2.1), an *Artifact Model* describes the behavior of an *Artifact* (*Artifact Behavioral Model*), and its interactions with other *Artifacts* and *Agents* (*Artifact Interaction Model*); however, as an Artifact is a re-active entity offering a set of services, the execution schedule of its *Activities* is always of the triggered type. In Figure 4 a part of the *Artifact Model* of the *Carrier* Artifact is reported.

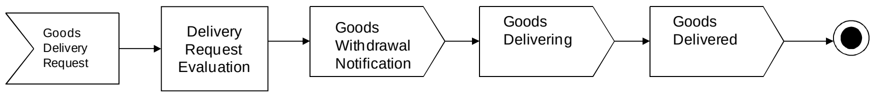
3.3 Simulation Design

Figure 5 shows a portion of the *Simulation Model* produced by adopting as the reference simulation framework the *Repast Simphony Toolkit* [15, 17]. In particular, Figure 5.a shows the organization of the Simulation Context (*SContexts*) whereas Figure 5.b shows the set of Simulation Behavior (*SBehavior*) of the Simulation Agent (*SAgent*) representing a *Vendor*. Specifically, for the *Vendor* three *SBehaviors* are defined, one for each *Activity* introduced in the *Agent Behavioral Model* during the Conceptual Modeling phase (see Figure 3.b). As an example, the *Order Planning SBehavior* in figure 5.b corresponds to the *Order Planning Activity* of the *Vendor Agent* reported in figure 3.b. The seamless transition between the two models is highlighted by the comparison between these two figures which manifests the straightforwardness of the mapping among the behavior of an Agent/Artifact, defined during the Conceptual Modeling phase in terms of *Activities* expressed by using the UML notation, and the behavior of an *SAgent*, defined during the *Simulation Design* phase in terms of *SBehaviors*.

- Carrier Activity Table -

Activity	Pre-cond.	Post-cond.	Execution schedule
Goods Delivering	…	…	Triggered

- UML Activity Diagram for the Goods Delivery Activity -

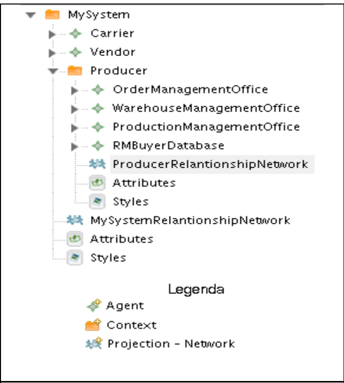


(b)** A part of the **Carrier Behavioral Model

Interaction	Activity	Initiator	Partners	Exchanged Information
Goods Delivery Request	Goods Delivering	Order Management Office	Carrier	Order Data
Goods Withdrawal Notification	Goods Delivering	Carrier	Warehouse Management Office	Withdrawal Date
Goods Delivering	Goods Delivering	Carrier	Vendor	Delivery Date
Goods Delivered	Goods Delivering	Carrier	Order Management Office	Delivery Date Delivery Status

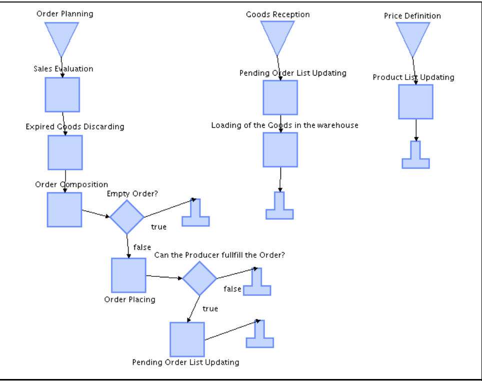
(c)** The **Carrier Interaction Model

Figure 4. A part of the Artifact Model of the Carrier Artifact.


MySystem
Carrier
Vendor
Producer
OrderManagementOffice
WarehouseManagementOffice
ProductionManagementOffice
RMBuyerDatabase
ProducerRelantionshipNetwork
Attributes
Styles
MySystemRelantionshipNetwork
Attributes
Styles

Legenda
Agent
Context
Projection - Network

(a) The Simulation Context



(b) The Simulation Behavior of the SAgent representing a Vendor

Figure 5. A part of the Simulation Model.

3.4 Simulation execution and result analysis

Starting from the *Simulation Model* described in the previous phase, great part of the simulation code is automatically generated by the *Repast Simphony Development Environment* [12], compiled by using a Java compiler and then loaded into the *Repast Simphony Runtime Environment* for the *Simulation Set-up* and *Execution*. In particular, according to the simulation objectives, the execution of the resulting Simulation Model made it possible to compare the three different considered production and pricing policies for the producer: *changeless*, *incremental*, and *adaptive*. With reference to Figure 6, which illustrates the diagram of the *profit* for the *Producer* Agent, it is possible to appreciate the great advantage given by the adoption of the *adaptive* production and pricing policy and how the *incremental* policy can lead to the complete failure of the enterprise if not opportunely corrected.

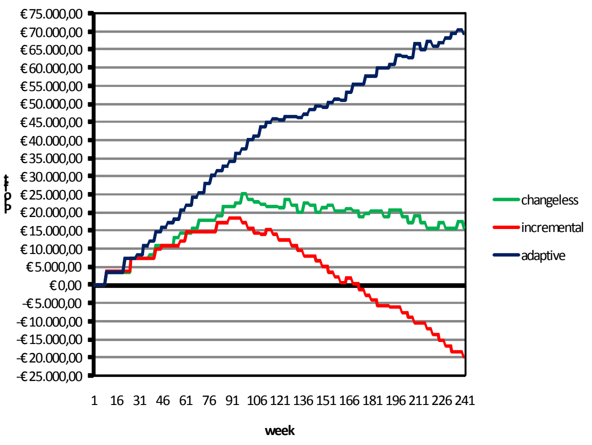


Figure 6. Profit for the Producer Agent under three different production and pricing policies.

4. CONCLUSIONS

To date, although several tools for ABMS are available, there are few methodologies and related processes which are able to cover all the phases from the analysis of the system under consideration to its modeling and subsequent simulation analysis. Moreover, the absence of visual modeling tools and techniques for ABMS often constitutes an entry barrier for whoever does not have advanced programming skills. To address these issues in this paper easyABMS, an integrated and iterative methodology for agent-based modeling and simulation of complex systems, has been presented along with a case study concerning the management of a three-stage supply chain which shows the effectiveness of the methodology in social and economic domains.

EasyABMS aims at supporting domain experts in fully exploiting the benefits of the ABMS while significantly reducing programming and implementation efforts, and represents a methodological approach capable to:

- guiding the domain experts from the analysis of the system under consideration to its modeling and simulation, as the phases which compose the process, the work-products of each phase, and the (seamless) transitions among the phases are fully specified;
- letting users concentrate their efforts on the modeling of the system and simulation analysis rather than the programming and implementation details, as the well-known Model Driven paradigm, in which the code is automatically generated from a set of (visual) models of the system, is adopted.

Currently, except for System Analysis and Conceptual System Modeling, all the phases of the process defined by easyABMS exploit the Repast Simphony Toolkit. Future research efforts will be devoted to: (i) extend the Repast Simphony Toolkit so to obtain an integrated ABMS environment which fully supports all the process phases; (ii) extensively experiment easyABMS in significant case studies concerning relevant social, financial, economic, and logistic issues; (iii) experiment the adoption of a meta-simulation framework for the Simulation Design phase so to obtain a Platform Independent Simulation Model which can be then translated into different platform-dependent simulation models.

5. REFERENCES

[1] C. Atkinson and T. Kühne. Model-driven development: A metamodeling foundation. IEEE Software, 20(5):36-41, 2003.

[2] G. Booch. Object-Oriented Analysis and Design with Applications. Addison-Wesley, 1994.

[3] A. Garro, W. Russo. An integrated agent-based process for the simulation of complex systems. Proceedings of the International Conference on Economic Science with Heterogeneous Interacting Agents (ESHIA), Warsaw, Poland, 19-21 June, 2008.

[4] A. Garro, W. Russo. An integrated and iterative process for agent-based modeling and simulation. Proceedings of the 5th International Conference of the European Social Simulation Association (ESSA), Brescia, Italy, 1-5 September, 2008.

[5] P. Giorgini, and B. Henderson-Sellers (Eds.). Agent-Oriented Methodologies, Idea Group Inc., 2005.

[6] T. Iba and N. Aoyama. Understanding Social Complex Systems with PlatBox Simulator. In Proc. of the 5th International Conference on Computational Intelligence in Economics and Finance (CIEF2006), pages 64-67, Taiwan, October 2006.

[7] T. Iba, Y. Matsuzawa, and N. Aoyama. From Conceptual Models to Simulation Models: Model Driven Development of Agent-Based Simulations. In Proc. of the 9th Workshop on Economics and Heterogeneous Interacting Agents. Kyoto, Japan, 2004.

[8] N. R. Jennings. An agent-based approach for building complex software systems. Commu-nications of the ACM, 44(4):35-41, 2001.

[9] MASON Home Page. George Mason University, Fairfax, VA, available at http://cs.gmu.edu/~eclab/projects/mason/.

[10] N. Minar, R. Burkhart, C. LAngton, and M. Askenazi. The Swarm Simulation System: A Toolkit for Building Multi-Agent Simulations. Working Paper 96-06-042. Santa Fe Institute, 1996.

[11] A. Molesini, A. Omicini, A. Ricci, E. Denti. Zooming multi-agent systems. In Proc of the 6th International Workshop on Agent-Oriented Software Engineering (AOSE 2005), AAMAS 2005, Utrecht, The Netherlands, 2005.

[12] M. J. North, T.R. Howe, N.T. Collier, and J.R. Vos. The Repast Simphony Development Environment. In Proc. of the

Agent 2005 Conference on Generative Social Processes, Models, and Mechanisms, Chicago, IL , October 2005.

[13] M. J. North, T.R. Howe, N.T. Collier, and J.R. Vos. Repast Simphony Runtime System. In Proc. of the Agent 2005 Conference on Generative Social Processes, Models, and Mechan-isms, Chicago, IL, October 2005.

[14] M. J. North, C. M. Macal. Managing Business Complexity: Discovering Strategic Solutions with Agent-Based Modeling and Simulation. Oxford University Press, 2007.

[15] M.J. North, E. Tatara, N.T. Collier, and J. Ozik. Visual Agent-Based Model Development with Repast Simphony. In Proc. of the Agent 2007 Conference on Complex Interaction and Social Emergence. Northwestern University, Evanston, IL, November2007.

[16] M. Resnick. Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Mi-croworlds. MIT Press, Cambridge, Mass., 1997.

[17] ROAD (Repast Organization for Architecture and Design). Repast Home Page, Chicago, IL, available as http://repast.sourceforge.net/.

[18] D. C. Schmidt. Model-Driven Engineering. IEEE Computer, 39(2):41-47, 2006.

[19] F. Strozzi, J. Bosch, and J.M. Zaldì. Beer game order policy optimization under changing customer demand. Decis. Support Syst., 42(4):2007, Elsevier Science Publishers B. V., Amsterdam, The Netherlands.

[20] Unified Modeling Language (UML) Specification. Version 2.1.2. Object Management Group Inc., 2007.

[21] U. Wilensky. NetLogo. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL, 1999.