easyABMS: A domain-expert oriented methodology for agent-based modeling and simulation

Alfredo Garro *, Wilma Russo

Department of Electronics, Computer and Systems Science (DEIS), University of Calabria, Via P. Bucci 41C, Rende (CS) 87036, Italy

**ABSTRACT**

Agent-Based Modeling and Simulation (ABMS) has arisen as new approach to effectively support domain experts to cope with the growing complexity of the problems which they have to face and solve. To date, few methodologies are available which can be exploited by domain experts with limited programming expertise to model and subsequently analyze complex systems typical of their application domains. In this paper the easyABMS methodology is proposed to overcome the lack of integrated methodologies able to seamlessly guide domain experts from the analysis of the system under consideration to its modeling and analysis of simulation results. The effectiveness of easyABMS is also experimented through a case study in the logistics domain which concerns the analysis of different policies for managing vehicles used for stacking and moving containers in a transhipment terminal.

© 2010 Elsevier B.V. All rights reserved.

1. Introduction

Agent-Based Modeling and Simulation (ABMS) is a new approach for analyzing and modeling complex systems [22] which is becoming acknowledged for its efficacy in several application domains (e.g. financial, economics, social sciences, logistics, engineering) [36]. ABMS, allows for the definition of a system model based on autonomous, goal-driven and interacting entities (agents) organized into societies which is then simulated so to obtain significant information on not only the properties of the system under consideration but also its evolution.

To date, ABMS models and their implementations for simulation environments can be obtained with varying degrees of difficulty and effectiveness by exploiting both available ABMS platforms [11,19,23,27–30,37,39,42,49] and agent-based methodologies [1,4,6,12,13,16,25,31,38,40,45,50,51].

While full-fledged ABMS platforms offer support suitable for simulation execution and subsequent results analysis, ABMS models obtained through direct implementation on a specific ABMS platform are of low abstraction level, platform dependent and, then, difficult to develop, verify, modify and update.

On the other hand, several agent-based methodologies make it possible to obtain agent models which are richer, both at micro (agent) and macro (organization) levels, although only a few of them provide simulation models which can be easily implemented on full-fledged ABMS platforms. The adaptation often required to overcome this issue is an additional effort which increases with the gap between the obtained agent-based models and the implementation models for ABMS simulation environments. In addition, the widespread adoption of a methodology by experts of typical ABMS domains, which often lack of advanced modeling and programming skills, demands for a domain-expert oriented methodology which should then be characterized by a fast learning curve and supported by visual modeling and simulation environments.
To address these issues, this paper presents a new domain-expert oriented methodology, easyABMS, which is specifically conceived for agent-based modeling and simulation of complex systems, and able to seamlessly cover all the phases from the analysis of the system under consideration to its modeling and the analysis of simulation results. easyABMS defines an iterative process which is integrated, model-driven and visual. In particular, each phase of the process refines the model of the system which has been produced in the preceding phase and the work-products obtained are mainly constituted by visual diagrams based on the UML notation. In addition, according to the model-driven paradigm, the simulation code is automatically generated from the derived system simulation model. On the basis of the simulation results, a new/modified and/or refined model of the system can subsequently be obtained through a new process iteration which can involve all or some process phases. Currently, easyABMS exploits the advanced features of visual modeling and of (semi)automatic code generation provided by the Repast Simphony Toolkit, a very popular and open source ABMS platform.

The effectiveness of easyABMS in supporting domain experts to fully exploit the benefits of the ABMS with a significant reduction of programming and implementation efforts is experimented through a case study in the logistics domain. Specifically, the case study is focused on the analysis of different policies for the management of vehicles used for stacking and moving containers (straddle carriers) in a container transhipment terminal.

The remainder of this paper is organized as follows: Section 2 presents the easyABMS methodology and the related process; Section 3 presents a brief introduction to the reference application domain (a container transhipment terminal) and its related management problems; Section 4 shows the application of easyABMS to the agent-based modeling and simulation of the straddle carrier routing and dispatching problem; Section 5 compares other related ABMS approaches to the easyABMS methodology; finally, conclusions are drawn and future works delineated.

2. The easyABMS methodology

The easyABMS methodology defines an iterative process for ABMS composed of seven subsequent phases (see Fig. 1) from the System Analysis to the Simulation Result Analysis. On the basis of the simulation results obtained a new iteration of the process which can involve all or some process phases can be executed for achieving new simulation objectives or those which have not yet been obtained. Specifically, the process phases are the following:

- **System Analysis**: in which the objectives of the simulation are specified and a preliminary understanding of the system and its organization is 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 established.

![Fig. 1. The reference process of the easyABMS methodology.](image-url)
– **Simulation Execution** and **Results Analysis**: in which the simulation results are analyzed with reference to the objectives of the simulation previously identified in the System Analysis phase.

Currently, all the simulation related phases are supported by the Repast Simphony Toolkit [37,43]. In particular, the **Simulation Design** and the **Simulation Code Generation** phases are supported by the Repast Simphony Development Environment [34], while the **Simulation Set-up**, the **Simulation Execution** and the **Simulation Results Analysis** phases are supported by the Repast Simphony Runtime Environment [35].

The models of the system generated by each process phase are produced according to the reference meta-model shown in Fig. 2. In particular, this meta-model, with reference to the main features presented in [15], is characterized by a single representation for entities, binary relationships with attributes to their ends, and a homogeneous and non-redundant representation for all the relationships. The meta-model structure results from the integration of three different parts which refer to the System Analysis, Conceptual System Modeling, and Simulation Design phase, respectively. Although these parts can be made self-consistent and exploited in isolation for specifying models at different abstraction levels, in the proposed meta-model, the concepts related to each phase are defined by extending and/or refining those of the previous phase; this allows for the seamless integration between the phases as the model produced in each phase extends and/or refines the model of the system produced in the previous phase.

The following sub-sections provide a brief description of each process phase and related work-products; more details will be provided through the case study in Section 4.

### 2.1. System Analysis

This phase is based on the principle of layering, exploits the well-known techniques of **Decomposition**, **Abstraction** and **Organization** [5,26] and is constituted of a sequence of **analysis steps**. In each step a new system representation is produced by applying the in-out **zooming mechanisms** [31] to the entities comprising the system representation obtained in the preceding analysis step. In the first analysis step, a starting abstraction level for analyzing the system is chosen.

According to the reference meta-model of the **System Analysis** phase (see Fig. 2), an **Entity** can be characterized by autonomous and goal-oriented behavior (**pro-active entity**), purely stimulus-response behavior (**re-active entity**), or can be **passive**. In addition, 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 entity evolution. Relationships can be either **intra-entity** (i.e. relationships among the component entities obtained by the zooming-in of an entity) or **inter-entity**.

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

![Fig. 2. The reference meta-model of the easyABMS methodology.](image-url)
identified entity (in terms of goals, capabilities and expected goal-directed behavior for pro-active entities, stimulus-response rules for re-active entities and description properties for passive entities) and the objectives of the simulation constitute the work-product of this phase (the Analysis Statement).

2.2. Conceptual System Modeling

In the Conceptual System Modeling phase, the Structural System Model is produced and, in particular, for each entity in the System Representation:

- The abstraction level suited to specific simulation objectives is chosen.
- The conceptual representation, in terms of Agent, Artifact or Society, is derived on the basis of the associations among the main concepts of the System Analysis and Conceptual System Modeling phases (see Fig. 2); specifically, composed, pro-active, reactive entities become Societies, Agents and Artifacts respectively; passive entities, which can be basically seen as resources, originate Artifacts acting as resource managers.
- The interactions with the other entities are obtained from the intra and inter-relationships where the latter cross the boundaries of societies.

The abstraction level chosen for an entity can be modified in successive iterations through which it is possible to produce new, modified, and/or refined Structural System Models.

For each entity in the produced Structural System Model a specific model is then defined, whose type can be one of the following depending on the entity type:

- **Society Model** which describes the entities which compose a Society, their type (Agent, Artifact, Society), the rules governing the Society (safety rules) and its evolution (liveness rules).
- **Agent Model** which details the complex goal of an Agent (Agent Goal Model), its behavior, as a set of periodically scheduled or triggered Activities (flow of Actions) which contribute to the achievement of the Agent goals (Agent Behavioral Model), and its interactions with other Agents and Artifacts in which the agent is involved (Agent Interaction Model).
- **Artifact Model** which describes the behavior of an Artifact as a set of triggered Activities related to the offered services (Artifact Behavioral Model), and its interactions with other Artifacts and Agents (Artifact Interaction Model).

2.3. Simulation Design

In this phase, starting from the Conceptual System Model a Simulation Model of the system, in terms of the abstractions offered by the framework exploited for the simulation is produced.

In Fig. 2 the basic concepts of the reference simulation framework (the Repast Symphony Toolkit [37,43]) are highlighted. Specifically, the central concept is the (simulation) Context (SContext) which represents an abstract environment in which (simulation) Agents (SAgents) can act and is provided with an internal state consisting of simple values and Data Fields (a n-dimensional field of values). In addition, an SContext can also support behaviors for the management of its internal state. SContexts can be organized hierarchically so to contain sub-SContexts which can have their own state. SAgents in an SContext can be organized by using Projections which are structures designed to define and enforce relationships among the SAgents in the SContext. In particular, a Network Projection defines the relationships of both acquaintance and influence between SAgents whereas Space Projections define (physical or logical) space structures (Grid, Scalar Fields, Continuous Space, Geography) in which the agents can be situated.

An SAgent can have multiple behaviors (SBehaviors), each operating on SAgent Properties and consists of a sequence of Steps; each Step can be associated with the execution of a Task or with the control of the flow of the Task execution (Loop, Join, Decision, End). Each SBehavior can be characterized by a Scheduled Method which defines a constant execution schedule, and by a Watch which periodically, on the basis of some watched parameters and conditions, triggers the execution of the behavior.

A Repast Symphony simulation model is defined by first specifying the structure and the characteristics of the root SContext and of all the possible nested sub-SContexts, in terms of their components (SAgents, Projections and sub-SContexts), and, then, specifying for each SAgent its Properties and SBehaviors, and for each SBehavior the component Steps, and the associated Scheduled Method and Watch.

The associations among the above described simulation concepts of the Repast Symphony Toolkit and the related concepts of the Conceptual System Model are reported in Fig. 2. The exploitation of these associations makes it possible to directly obtain, starting from the Conceptual System Model, the Simulation Model of the System as follows:

- 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.
- Artifacts and Agents become Repast Simulation Agents (SAgents), the Activities which constitute their behaviors are easily converted into Repast Simulation Behaviors (SBehaviors).
- 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 [2,44], the Repast Symphony Development Environment [34] is able to automatically generate a great part of the simulation code from the derived Simulation Model of the system. The simulation code which can be extended with additional Java and XML code is then compiled by the Repast Symphony Development Environment using a Java compiler and loaded into the Repast Symphony Runtime Environment.

The simulation executed by the Repast Symphony Runtime Environment can start after setting-up: (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.

Finally, the obtained simulation results can also be analyzed by exploiting the analysis tools (Matlab, R, VisAd, iReport, Jung) which can be directly invoked from the Repast Symphony Runtime Environment so to verify whether the objectives of the simulation identified during the System Analysis phase have been achieved. Where objectives have not been achieved or where new simulation objectives emerge, a new iteration of the process can be executed, which can then involve all or some process phases so that new/modified and/or refined models of the system can be produced for achieving the remaining/new simulation objectives.

3. Management of a container transhipment terminal

The effectiveness of easyABMS in supporting domain experts to fully exploit the benefits of the ABMS with a significant reduction of programming and implementation efforts is demonstrated through a case study in a typical ABMS domain which is introduced in this Section. In particular, the maritime transportation domain is chosen as it is becoming a crucial asset in the global economy due to the continuous growth in the volume of goods exchanged around world further boosted by the rising Chinese and Indian economies.

The current maritime transportation system is based on a hub and spoke model [46] whereby ultra-large containerships operate between a limited number of mayor (mega)transhipment terminals (hubs), and smaller vessels (feeders) which link the hubs with other minor ports (spokes). In this scenario, a hub terminal must maintain a high level of efficiency, not only to avoid traffic congestion but also to increase its competitiveness as some main characteristics (geographical, structural and technological) which also determine the competitiveness of a container terminal can be modified only on a long term perspective.

Thus, it becomes crucial to increase hub efficiency, rendering it more competitive through the optimal management of terminal resources and optimizing tactical and operational logistics.

In the next sub-section, the organization of a maritime container terminal and some primary management issues are briefly discussed; a more complete description can be found in [32].

3.1. Organization of a container transhipment terminal

Each ship approaching a maritime terminal enters in a harbour and waits to moor at an assigned berth position along the terminal quay which is equipped with giant cranes (quay cranes) for loading and unloading containers. These containers, in a Direct Transfer System (DTS) terminal, are transferred to and from the terminal yard by a fleet of vehicles (straddle carrier) which are able to stack containers in the yard. In contrast, in an Indirect Transfer System (ITS) terminal, containers are moved by trucks and trailers from the quay to the yard and vice versa and staked by yard cranes.

In this context, the main logistic processes and related management problems can be grouped in relation to the flow of containers in the terminal as shown and briefly described in Table 1; other issues are related to inter-terminal transportation and to possibly link with other transportation modes. Moreover, a transversal issue is related to the human resources management [32].

<table>
<thead>
<tr>
<th>Phase</th>
<th>Problem</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arrival of the containership</td>
<td>Quay crane assignment problem (QCAP)</td>
<td>Determining the number of quay cranes to assign to an incoming vessel</td>
</tr>
<tr>
<td></td>
<td>Berth allocation problem (BAP)</td>
<td>Assigning incoming ships to berths, by taking into account constraints in both spatial and temporal dimensions so to minimize the time each ship spends in port (turnaround time)</td>
</tr>
<tr>
<td>Unloading and loading of the ship</td>
<td>Quay crane scheduling problem (QCSP)</td>
<td>Determining a sequence of unloading and loading movements for cranes assigned to a vessel in order to minimize the vessel completion time as well as the crane idle time</td>
</tr>
<tr>
<td>Transport of containers from the ship to the yard and vice versa</td>
<td>Yard management</td>
<td>Allocating and reallocating the containers in the yard in order to reduce the amount of time required to handle of each vessel</td>
</tr>
<tr>
<td></td>
<td>Straddle carrier routing and dispatching (SCRD)</td>
<td>Determining the operation to be performed by the straddle carries to maximize the productivity of each crane</td>
</tr>
</tbody>
</table>
These very fundamental issues are not only reciprocally related, but the large-scale nature of hub management makes the use of standard exact solution algorithms impractical. In fact, the management of such large and intricately complex systems requires new modeling methods which must also generate proof-of-concept simulations.

In the following Section, the effectiveness of the ABMS approach and the easyABMS methodology is shown focusing on the straddle carrier routing and dispatching problem (SCRDP) [33]; with reference to the different management problems in a container transhipment terminal (see Table 1), a more complete and domain specific agent-based simulator has been proposed in [21].

4. Modeling and simulating straddle carrier routing and dispatching through easyABMS

4.1. System Analysis

The main indicator of optimal performance in a container transhipment terminal is the average ship-turn-around time which is the average time-lapse between a ship’s arrival and its departure, starting from the amount of time the ship waits for a berth (berth waiting time) and the duration for which the ship is docked for unloading and loading operations (handling time). In the following, the focus is set on the handling time which is highly dependent on the productivity of the quay cranes (QCs) and, as a consequence, on the management policies of the straddle carriers (SCs). Moreover, lower handling times can also reduce berth waiting times as it is possible to sooner release berths for incoming ships.

Specifically, to maximize the productivity of the QCs in a DTS container terminal, the SCs should operate so that the buffer of each crane, which has a limited capacity of only a few containers, is not full/not empty if the crane is performing the discharging/loading phase. Specifically, there are two main policies for organizing the work of SCs:

– Dedicated modality: a given number of SCs are allocated to each QC to follow its working phases.
– Shared modality (or pooling): a group of SCs is shared by two or more QCs which work on the same ship or on adjacent berthed ships and, possibly, frequently swapping between the tasks of loading and discharging containers.

The shared modality presents several benefits with respect to the dedicated mode: (i) reduction in the number of empty trips done by the SCs (i.e. travels without carrying any container), as the SCs can fruitfully alternate between trips carrying containers from the yard to the cranes which are loading outgoing cargo and trips back to the yard, carrying discharged cargo; (ii) more constant value of productivity of both QCs and SCs as, when a crane is not working, the SC of a pool can speed up operations of the other QCs.

A quantitative evaluation of the aforementioned benefits is not easy to obtain through traditional analytical models. Moreover, classical dispatching models [33] often fail to provide dynamic assignment of container moves to SCs of a pool in order to speed up the loading/discharging operations (the straddle carriers pooling problem – SCPP). To overcome these shortcomings, an agent-based model can be defined and simulated with the following main objectives:

(i) Quantifying the benefits of the pooling modality with reference to system productivity (vessels handling time) and cost reduction (numbers of exploited SCs and total distance covered).
(ii) Obtaining an effective solution for the dynamic assignment of container moves to the SCs of a pool which can be used for automatically drive the coordinated behavior of the SCs in a real container terminal.

Fig. 3. System Representation.
The System Representation obtained on the basis of the identified simulation objectives is reported in Fig. 3. All the entities represented in Fig. 3 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.

4.2. Conceptual System Modeling

The Structural System Model derived from the System Representation is reported in Fig. 4; in particular, as the simulation objectives concern management policies of SCs, the level of representation chosen for the Vessel is more abstract with respect to the level resulting from the System Analysis phase.

For each entity in the Structural System Model the corresponding Society, Agent or Artifact Model is defined (see Section 2.2). As an example, the following sub-sections report the Society Model for the Container Terminal Society, the Agent Model for the Straddle Carrier Agent and the Artifact Model for the Movement Task Assigner Artifact.

4.2.1. The Container Terminal Society Model

The Society Model of the Container Terminal Society is shown in Fig. 5 which reports the different entities which compose the Society, the safety and liveness rules which govern it and its dynamics.

4.2.2. The Straddle Carrier Agent Model

Part of the Agent Model of the Straddle Carrier Agent is shown in Fig. 6 and in particular:

---

**Fig. 4.** Structural System Model.

---

**Fig. 5.** A part of the Society Model of the Container Terminal Society.

---

<table>
<thead>
<tr>
<th>Entity</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vessel</td>
<td>Artifact (Resource Manager)</td>
</tr>
<tr>
<td>Quay Crane (QC)</td>
<td>Agent</td>
</tr>
<tr>
<td>Buffer</td>
<td>Artifact (Resource Manager)</td>
</tr>
<tr>
<td>Straddle Carrier (SC)</td>
<td>Agent</td>
</tr>
<tr>
<td>Movement Task Assigner</td>
<td>Artifact</td>
</tr>
<tr>
<td>Yard</td>
<td>Artifact (Resource Manager)</td>
</tr>
</tbody>
</table>

**Safety rules**

\[ S_{\text{Term}1}. \quad \text{NC}_{vi}(t) = \text{NC}_{vi}(t_0) - \text{NCD}_{vi}(t) + \text{NCL}_{vi}(t); \]
where \( \text{NC}_{i}(t) \) is the number of containers on the Vessel \( i \) at time \( t \); \( \text{NCD}_{vi}(t) \) is the number of containers that have been discharged from the Vessel \( i \) up to time \( t \); \( \text{NCL}_{vi}(t) \) is the number of containers that have been loaded onto the Vessel \( i \) up to time \( t \).

\[ S_{\text{Term}2}. \]

**Liveness rules**

\[ L_{\text{CTerm}1}. \quad \text{A Quay Crane cannot download a container on its buffer if the buffer is full.} \]

\[ L_{\text{CTerm}2}. \quad \ldots \]
The Straddle Carrier Goal Model (Fig. 6a) in which, as the two goals (Movement of containers from Buffer to Yard and Movement of containers from Yard to Buffer) can be achieved independently, no achievement relationships are present.

A part of the Straddle Carrier Behavioral Model (Fig. 6b); in particular, the Straddle Carrier Activity Table specifies the activities (Container Movement Activity) which the Straddle Carrier Agent executes for achieving its goals, along with the pre and post conditions and the execution schedule (periodical). Moreover, as the definition of an Agent Behavioral Model requires that each activity in the Agent Activity Table must be further described by an UML [47] Activity Diagram, the diagram for the Container Movement Activity is also shown. The UML Activity Diagram must be further enriched with an Activity Action Table (not shown in figure due to space limitations) which reports, for each single component 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).

(c) The Straddle Carrier Interaction Model

Fig. 6. A part of the Agent Model of the Straddle Carrier Agent.
4.2.3. The Movement Task Assigner Artifact Model

Part of the Artifact Model of the Movement Task Assigner Artifact is shown in Fig. 7, and, in particular, the part of the Movement Task Assigner Behavioral Model which describes the Task Assignment Activity triggered by an SC requesting a new container movement to be performed. Specifically, at the completion of its container movement the SC requests the next assignment from the Movement Task Assigner (see Fig. 6c). The Movement Task Assigner must then decide, from available

---

**Table: Movement Task Assigner Activity Table**

<table>
<thead>
<tr>
<th>Activity</th>
<th>Service</th>
<th>Preconditions</th>
<th>Post conditions</th>
<th>Execution Schedule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task Assignment</td>
<td>Movement Task Assignment</td>
<td>A movement task must be available unless the Vessel handling is completed</td>
<td>If available, a new movement task must be assigned to the SC</td>
<td>Triggered</td>
</tr>
</tbody>
</table>

**Diagram: UML Activity Diagram for the Task Assignment Activity**

- [Vessel Handling completed]
  - Task Assignment Request
    - [Vessel Handling not completed]
      - Evaluate next moves for the other SCs in the pool
      - Assign a move to the requesting SC

**Fig. 7.** A part of the Movement Task Assigner Behavioral Model.

---

- The Straddle Carrier Interaction Model (Fig. 6c) which specifies, for each action of the interaction type (Task Assignment Request, Assigner Response) of the Container Movement Activity, the initiator, the partners of the interaction and the exchanged information.

**Diagram: Simulation Context**

**Diagram: Container Movement SBehavior of the Straddle Carrier SAgent**

**Fig. 8.** A part of the Simulation Model.
moves, the next best move for the requesting SC taking into account also subsequent moves which could be assigned to the other SCs in the pool (Lookahead Policy). Such planning could be dynamically revised at the next task assignment request.

4.3. Simulation Design

Fig. 8a–b show a portion of the Simulation Model produced by adopting the Repast Simphony Toolkit [37,43] as the reference simulation framework. Fig. 8a shows the organization of the Simulation Context (SContext) whereas Fig. 8b shows a Simulation Behavior (SBehavior) of the SAgent representing a Straddle Carrier. In particular, the Container Movement SBehavior in Fig. 8b corresponds to the Container Movement Activity reported in Fig. 6b. The seamless transition between the two models is highlighted by the comparison between these two figures which clearly demonstrates that the behavior of an Agent/Artifact, defined during the Conceptual Modeling phase in terms of Activities expressed by using the UML notation, can be directly mapped onto that of an SAgent, defined during the Simulation Design phase in terms of SBehaviors.

4.4. Simulation Set-up, Execution and Results Analysis

Starting from the Simulation Model a great part of the simulation code is automatically generated by the Repast Simphony Development Environment [34], compiled by using a Java compiler and then loaded into the Repast Simphony Runtime Environment for the Simulation Set-up and Execution.

According to the simulation objectives, the execution of the resulting Simulation Model made it possible to compare and quantify the benefits of both dedicated and pooling modalities. In particular, several simulations have been executed for different scenarios in order to evaluate: the Quay Crane Idle Time (QCIT), the Straddle Carrier Covered Distance (SCCD), and the Straddle Carrier Idle Time (SCIT). As an example, Fig. 9a–b illustrate the QCIT and the SCCD, in the two different modalities, with reference to a simulation scenario based on real-life organizational topology and equipment typologies of the Gioa Taurro Container Terminal [11] which is one of the largest hub port in the Mediterranean Sea and the reference one for the industrial research project PROMIS (logistic PROcess Management and Intelligence System). In this simulation scenario one Vessel is handled by two QCs for the loading and discharging of 50 containers respectively according to provided loading/discharging plans, other main simulation set-up parameters are reported in Table 2. The results shown in Fig. 9, which are results

| Table 2 |
| Some main simulation set-up parameters. |

| Yard parameters | Number of yard blocks 16 |
| Length of a block (m) 150 |
| Width of a block (m) 100 |
| Number of container slots in a block 16 |
| Number of container lines in a block 32 |
| Number of container layers in a block 3 |
| Distance between adjacent blocks (m) 15 |

| Quay crane parameters | Average time for a quay crane move (s) 67 |
| Average repositioning time between two subsequent quay crane moves (s) 33 |
| Number of container slots in a quay crane's buffer 5 |

| Straddle carrier parameters | Average speed of a loaded straddle carrier (m/s) 3.7 |
| Average speed of an empty straddle carrier (m/s) 4.7 |
| Average time required for a straddle carrier to load/unload a container (s) 120 |

Fig. 9. Some simulation results.
averaged from 30 simulation runs, made it possible to quantify the significant advantage of the pooling modality in terms of vessel handling time and cost reduction.

5. Related work

For the specifications of ABMS models and/or their implementations in specific simulation environments, even if with varying degrees of effectiveness and required effort, several ABMS tools and platforms, agent modeling languages and agent-based methodologies could be exploited. Specifically, ABMS tools and platforms [10,19,23,27–30,37,39,42,49], which usually make available visual editors for defining ABMS models and, in particular, for specifying agent behaviors, as well as semi-automatic code generation capabilities, actually provide satisfactory support for only simple models. Complex system modeling which indeed requires sizeable extensions of the basic behavioral templates provided by the tools demands significant programming skills. In addition, as these tools do not refer to specific ABMS methodologies, system models are usually obtained by extending and refining available examples and case studies. This makes such platform-dependent models of low level of abstraction and flexibility.

Agent modeling languages [3,9,20,48] which mainly come from the agent-oriented software engineering (AOSE) domain, and enable for a clear, high level and often semantically well-founded definition of ABMS models, allow for the definition of richer agent models at both micro (agent) and macro (organization) levels than that of the ABMS simulation toolkits; nevertheless advanced modeling skills are required. Moreover, as agent modeling languages do not usually refer to any ABMS platform, transitions between the obtained design models and the operational models for available ABMS platforms are quite difficult to perform due to the significant gap between them. Therefore, approaches based on only ABMS tools and platforms or agent modeling languages are not very effective for fully supporting modeling and simulation of complex systems.

Agent-oriented methodologies can overcome such limitations. In fact, several methodologies introduced in the context of the AOSE, such as PASSI [12], PASSIM [13], ADELFE [4], GAIA [50] and GAIA2 [51], TROPOS [6], SONIA [1], SODA + Zoom [31], MESSAGE [8], INGENIAS [40], O-MaSE [16], SADDE [45], Prometheus [38], and MAS-Common-KADS [25], could be considered as they provide processes, techniques and/or abstractions to fruitfully exploit for ABMS. However, none of these methodologies allow domain experts with limited programming skills to obtain agent-based models which can be directly and effortlessly executed on full-fledged ABMS environments able to fully support the phases of simulation and result-analysis. In fact, the adaptation between the models obtained and the target simulation models requires significant efforts which are time-consuming, error-prone and demands advanced programming skills. As a consequence, only methodologies specifically conceived for ABMS or ABMS extensions of available AOSE methodologies will be discussed and compared with reference to the following identified main features (see Table 3):

- **Completeness**: a process should cover all the phases, from the analysis of the system under consideration to its modeling and simulation.
- **Integration**: the models of the system produced in each process phase should be a seamless and well-defined refinement of those obtained in the preceding phases (i.e. the gap among models of subsequent phases should be limited and effortless to fill).
- **Domain-expert orientation**: the methodology should be made user-friendly for domain experts, not requiring advanced modeling and programming skills, be mastered quickly with a fast learning curve and adequately supported by visual modeling and ABMS simulation environments.

Among the ABMS methodologies which are extensions of existing AOSE methodologies an interesting proposal is presented in [41] where an extension of the well-known INGENIAS methodology [40] is specifically conceived for modeling social systems and based on model-driven engineering practices. To specify complex social systems this extension relies on the INGENIAS modeling language which is structured in five packages: Organization, Agent, Goals-Tasks, Interactions and Environment. The INGENIAS Development Kit then facilitates the editing of the models and the definition of transformation rules from the INGENIAS-based social system specifications to platform dependent simulation models, thus enabling automatic code generation which is currently provided for Repast J [36].

In [24] the authors have proposed a process for ABMS which is based on Model-Driven Development and consists of three major phases: the Conceptual Modeling Phase, the Simulation Design Phase and the Verification Phase. In the Conceptual Modeling Phase a conceptual model of the system under analysis consisting of a Conceptual-Model Class Diagram, a set of Activity Diagrams and a Communication-Sequence Diagram is produced according to the BEFM (Boxed Economy Foundation Model) framework. In the Simulation Design Phase, a simulation model of the system, consisting of an executable program for the Boxed Economy Simulation Platform (BESP), is obtained from the produced conceptual model by exploiting the Foundation Model Framework (FMFW). The work-products of this phase are a Simulation-Model Class Diagram, a set of Statechart Diagrams, an Initial World-Settings and the corresponding code. In the Verification Phase, the simulation is executed for analyzing the simulation results and, if necessary, a new iteration of the process starting from the first or the second phase can be conducted.

In [7] the authors described the MALEVA agent model, an abstract model for constructing and implementing operational models for multi-agent based simulations; in particular, the approach is based on an incremental design and the building of
Table 3
Comparison of available methodologies for ABMS.

<table>
<thead>
<tr>
<th>Methodology</th>
<th>Completeness (supported process phases)</th>
<th>Integration (gap to fill among models of subsequent phases)</th>
<th>Domain-expert orientation</th>
<th>Domain model</th>
<th>Simulation environment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>System Analysis (P1)</td>
<td>Application domain</td>
<td>Social systems</td>
<td>UML-based</td>
<td>INGENIAS Development Kit</td>
</tr>
<tr>
<td>INGENIAS</td>
<td>X</td>
<td>Limited gap from P1 to P2 and from P2 to P3</td>
<td>Advanced</td>
<td>Medium</td>
<td>Slow</td>
</tr>
<tr>
<td>Boxed</td>
<td>X</td>
<td>Partially supported</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Economy</td>
<td>X</td>
<td>Limited gap from P1 to P2 and from P2 to P3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>Limited gap from P1 to P2 and from P2 to P3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MALEVA</td>
<td>X</td>
<td>Partially supported</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>easyABMS</td>
<td>X</td>
<td>Significant gap from P1 to P2</td>
<td>Specific</td>
<td>Medium</td>
<td>CGraphGen</td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>Seamless transition from P1 to P2 and from P2 to P3</td>
<td>UML-based</td>
<td>Medium</td>
<td></td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>Seamless transition from P1 to P2 and from P2 to P3</td>
<td>UML-based</td>
<td>Medium</td>
<td></td>
</tr>
</tbody>
</table>

*Boxed Economy Simulation Platform Not provided*
agent behaviors or activities by composing simpler behaviors/actions, reified as software components. Therefore, the authors assume that there is a library of pre-defined behaviors associated with the target application domains. A component may be primitive (written in a programming language such as Java) or composite. A specific characteristic of the MALEVA model is the distinction between the activation control flow and the data flow which renders components independent of their activation logic and therefore more reusable.

Finally, easyABMS, which is presented in this paper, is a model-driven methodology specifically conceived for ABMS and provides a process which is both complete, as it covers all phases from that of the analysis of the system under consideration to those of the modeling and simulation analysis, as well as integrated, since each phase refines the model of the system, mainly composed of visual diagrams based on the UML notation, which has been produced in the preceding phase (see Section 2). In addition, the simulation model obtained in the Simulation Design phase enables the automatic code generation for the Repast Simphony Platform [34] and the subsequent exploitation of the related simulation and results analysis environment.

With reference to the discussed methodologies, the approaches which better fulfill the essential requirements for an effective ABMS methodology (i.e. completeness, integration and domain-expert orientation) are Boxed Economy, INGENIAS for ABMS and easyABMS (see Table 3). These differ with respect to their domain-expert orientation features which are related to:

- **Application domain**: general purpose for Boxed Economy and easyABMS, specific (social-system oriented) for INGENIAS for ABMS.
- **Modeling skills**: low for easyABMS, medium for Boxed Economy, advanced for INGENIAS for ABMS, as determined from both the variety of modeling notation to use and number of diagrams to draw.
- **Programming skills**: medium for INGENIAS for ABMS and easyABMS, advanced for Boxed Economy, as determined by the different support offered for generating the simulation code for the target ABMS platform.
- **Learning curve**: medium for Boxed Economy and easyABMS, slow for INGENIAS for ABMS due to the manifold and variegated aspects of the system that the methodology must deal with.
- **Simulation environment**: proprietary for Boxed Economy, Repast J for INGENIAS for ABMS, Repast Simphony for easyABMS.

While it is widely agreed that there are significant benefits to be gained from exploiting open source, popular and really adopted ABMS modeling and simulation environments (as is in INGENIAS for ABMS and easyABMS), whether and how far a fast learning curve and lower levels of modeling and programming skills which are required (as is for easyABMS) could promote a widespread adoption of a given methodology by domain experts is a question that can be answered only after extensive experimentations involving experts in a variety of application domains.

### 6. Conclusions and future work

Several tools for ABMS are now available along with methodologies for the development of agent-based systems mainly proposed in the context of Agent-Oriented Software Engineering (AOSE). Nonetheless, few proposals are available which integrate the methodological features coming from the AOSE with the modeling and simulation features of modern ABMS tools. As a result, there is little support for domain experts with limiting programming expertise in the whole process which goes from the system analysis to the analysis of simulation results. To address these issues, this paper has presented easyABMS, a full-fledged methodology for the agent-based modeling and simulation of complex systems which fruitfully exploits both AOSE modeling techniques and simulation tools specifically conceived for ABMS. The effectiveness of easyABMS has been shown using a case study in the logistics domain which concerns the analysis of different policies for managing Straddle Carriers in a Container Transhipment Terminal. In particular, easyABMS overcomes the main limitations of classical analytical models and easily provides a quantitative assessment of two primary Straddle Carrier management policies and an effective solution in guiding the dynamic assignment of container moves. easyABMS demonstrated able to seamlessly guide domain experts from the analysis of the system under consideration to its modeling and simulation as the phases which compose the easyABMS process, the work-products of each phase and the (seamless) transitions among the phases are fully specified. In addition, easyABMS focuses on system modeling and simulation analysis rather than details related to programming and implementation as it exploits the model-driven paradigm, making it possible the automatic code generation from a set of (visual) models of the system.

Future research efforts will be devoted to: (i) extending the Repast Simphony Toolkit so to obtain an integrated and open source ABMS environment which fully supports all the process phases also comprising the System Analysis and Conceptual System Modeling phases; (ii) extensively evaluate the effectiveness of easyABMS using case studies from contexts involving social, financial, economic, and logistics systems; (iii) adopting a meta-simulation framework for the Simulation Design phase so to obtain a platform-independent simulation model which can then be translated into different platform-dependent simulation models.
References


