A MODEL-DRIVEN ARCHITECTURE APPROACH FOR AGENT-BASED MODELING AND SIMULATION

Alfredo Garro, Francesco Parisi and Wilma Russo
Department of Electronics, Computer and Systems Science (DEIS)
University of Calabria, Via P. Bucci 41C, Rende (CS), 87036, Cosenza, Italy

Keywords: Agent-Based Modeling and Simulation, Model Driven Development, Model Driven Architecture.

Abstract: It is widely agreed that a widespread adoption of the Agent-Based Modeling and Simulation (ABMS) approach by experts of typical ABMS domains demands for well-defined processes, modeling techniques and tools able to fully support them in modeling and simulating complex systems. To this end, the paper proposes a Model-Driven process which conforms to the OMG Model-Driven Architecture (MDA) and enables the definition of Platform-Independent simulation Models from which Platform-Dependent simulation Models and the related code can be automatically obtained with significantly reduced programming and implementation efforts.

1 INTRODUCTION

Agent Based Modeling and Simulation (ABMS) represents a new and powerful approach for analyzing and modeling complex systems in a wide range of application domains (e.g. financial, economic, social, logistic, chemical, engineering) (Garro, 2009). In fact, ABMS can fully represent a system at different levels of complexity through the use of autonomous, goal-driven and interacting entities (agents) organized into societies which exhibit emergent properties. The agent-based model of a system can then be executed to simulate the behavior of the complete system so that knowledge of the behaviors of the entities (micro-level) produce an understanding of the overall outcome at the system-level (macro-level).

To date, exploiting the currently available ABMS platforms (North, 2007), agent-based simulation models can be obtained using two main approaches: (i) direct implementation on a specific ABMS platform: this approach inevitably suffers from the limitations and particular features of the chosen platform; (ii) manual adaption of a conceptual system model, possibly obtained by exploiting an AOSE (Agent-Oriented Software Engineering) methodology (Henderson-Sellers, 2005), to a specific ABMS platform: this requires additional adaptation efforts, the magnitude of which increases depending on the gap between the conceptual and implementation models of the system. In fact, both these approaches require significant implementation efforts and lead to agent-based simulation models which are at a low-abstraction level, strongly platform-dependent, and therefore not easy to verify, modify and update (Garro, 2010), (Iba, 2004), (Nebrijo Duarte, 2009). On the other hand, although there is an increasing interest in the definition of Platform-Independent Simulation Models (AMP, 2011), (North, 2007) which enable the exploitation of more high-level simulation design abstractions and the automatic code generation for different target simulation environments, there is a lack of process able to guide and support ABMS practitioners in the definition of these simulation models starting from a conceptual, and domain-expert-oriented modeling of the system without taking into account simulation configuration details.

To address these issues, this paper aims to extend the benefits of an emerging software engineering approach, the Model-Driven Development (MDD) (Atkinson, 2003), to ABMS practitioners by proposing a Model-Driven approach for ABMS which conforms to the OMG Model-Driven Architecture (MDA) (OMG, 2010) and then allows to (automatically) produce *Platform-Dependent simulation Models* starting from a

Garro A., Parisi F. and Russo W..
A MODEL-DRIVEN ARCHITECTURE APPROACH FOR AGENT-BASED MODELING AND SIMULATION.
DOI: 10.5220/0003648400740083
In *Proceedings of 1st International Conference on Simulation and Modeling Methodologies, Technologies and Applications* (SIMULTECH-2011), pages 74-83
ISBN: 978-989-8425-78-2

Platform-Independent simulation Model obtained on the basis of a preliminary *Computation Independent Model*. Specifically, Platform-Independent simulation Models are produced by exploiting the AMF framework defined in the AMP (Agent-Modeling Platform) Eclipse Project which currently represents the only effort able to provide automatic generation of Platform-Dependent simulation Models and related code for several ABMS platforms (AMP, 2011).

The remainder of this paper is organized as follows: the proposed MDA-based approach for ABMS is presented (Section 2) and then exemplified (Section 3) with reference to a popular problem (the *Demographic Prisoner's Dilemma*) able to represent several social and economic scenarios. Finally, conclusions are drawn and future works delineated.

2 AN MDA-BASED APPROACH FOR ABMS

The Model-Driven Development (MDD) approach conceives system development in terms of a chain of model transformations (Atkinson, 2003). The most mature MDD proposal is the Model-Driven Architecture (MDA) (OMG, 2010) launched by the Object Management Group (OMG) which defines three main abstraction levels of a system and the resulting model transformations: (i) a *Computation Independent Model* (CIM) which describes context, requirements and organization of a system at a conceptual level; (ii) a *Platform-Independent Model* (PIM) which specifies architectural and behavioral aspects of the system without referring to any specific software platform; (iii) *Platform-Specific Models* (PSMs) which describe the realization of the system for specific software platforms and from which code and other development artifacts can be straightforwardly derived. Transformations between models (M1 Layer) are enabled by both the *metamodels* (M2 Layer) which they conform to and the mappings among these metamodels which must be defined as instances of the *meta-metamodel* (M3 Layer) represented by the Meta Object Facility (MOF) (OMG, 2006) (see Figure 1). Thus, given a source model and the mapping between its metamodel and the target metamodel, the target model can be generated.

The MDA proposal can be effectively exploited in the ABMS domain to obtain platform-independent agent-based models which are not only easy to verify, modify and update but also require significantly reduced programming and implementation efforts. However, to map the basic MDA concepts, which have been specifically conceived for the Software Engineering domain, into the ABMS counterparts, the following components must be available: (i) a reference CIM metamodel for the definition of CIMs which supports the agent-based conceptual system modeling carried out through both abstract and domain-expert oriented concepts; (ii) a PIM metamodel for the definition of Platform-Independent ABMS Models; (iii) a PSM metamodel for each target ABMS simulation platform; (iv) mappings among these metamodels so to enable ABMS model transformations.

The main effort for the definition of Platform-Independent ABMS Models, is currently represented

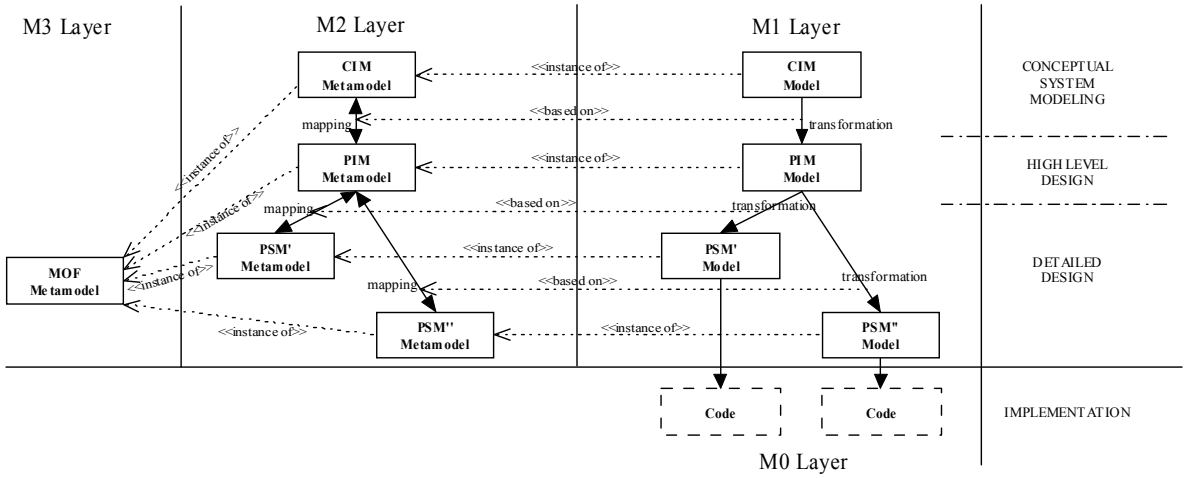


Figure 1: The MDA-based Process.

by AMF (AMP, 2011) which provides the automatic generation of PSMs and the related code for three popular ABMS platforms (AMP, 2011) (North, 2005) (Parker, 2001). Therefore, to enable the definition of complete Model-Driven ABMS processes which conform to the MDA-based process depicted in Figure 1a, a CIM metamodel with the above highlighted features (Section 2.1), an AMF-based PIM metamodel (Section 2.2), and the mapping between the two defined metamodels (Section 2.3) are provided.

2.1 The CIM Metamodel

The CIM metamodel is defined by adopting a light, task-based model of agents' behaviors which combines the strengths of several well-known task-based agent models (Bernon, 2004). This metamodel is quite general and plain, as required by the abstraction level for which it has been conceived, but powerful enough for representing, at a conceptual level, a great variety of systems in typical ABMS domains (e.g. financial, economic, social, logistic, physical science, engineering).

In particular, the CIM metamodel reported in Figure 2 is centered on the concept of *Agent*. An *Agent*, which is situated in an *Environment* constituted by *Resources,* is characterized by a *Behavior* and a set of *Properties*. *Agents* can be organized into *Societies* which in turn can be organized in *sub-societies*. A *Behavior* is composed by a set of *Tasks* organized according to *Composition Task Rules*. Each *Task*, which can act on a set of environment's *Resources*, is structured as an UML Activity Diagram which consists of a set of linked *Actions* that can be either *Control Flow* (pseudo) actions (i.e. *start*, *end*, *split*, *join*, *decision*, *merge*, *sequence*) or *Computation* and *Interaction* actions (i.e. *outgoing* or *incoming signals*) (OMG, 2009).

2.2 The PIM Metamodel

The PIM metamodel is derived from the AMF framework (AMP, 2011) as AMF represents the most significant effort towards the definition of Platform-Independent simulation Models. Specifically, these PIM models can be defined through a hierarchical visual editor and represented by XML documents (Schauerhuber, 2006) which are exploited for the generation of PSMs and related code which currently targets the Ascape (Parker, 2001), Escape (AMP, 2011) and Repast Simphony simulation platforms (North, 2005).

The derived AMF-based PIM metamodel (see Figure 3) is centered on the concept of (Simulation) *Context* (*SContext*) which represents an abstract environment in which (Simulation) *Agents* (*SAgents*) can act. An *SAgent* is provided with an internal *state* consisting of a set of *SAttributes*, a visualization style *SStyle*, and a group of *AActs* (*AGroup*) which constitute its behavior. An *AAct* is characterized by an *Execution Setting* which establishes when its execution can start, its periodicity and its priority.

SContexts, which are themselves *SAgents*, can be organized hierarchically and contain *sub-SContexts*. *SAgents* in an *SContext* can be organized by using *SProjections* which are structures designed to define and enforce relationships among *SAgents* in the *SContext*. In particular, a *SNetwork* projection defines the relationships of both acquaintance and influence between *SAgents* whereas *SGrid, SSpace, SGeography* and *SValue Layer* projections define either the physical space or logical structures in which the agents can be situated.

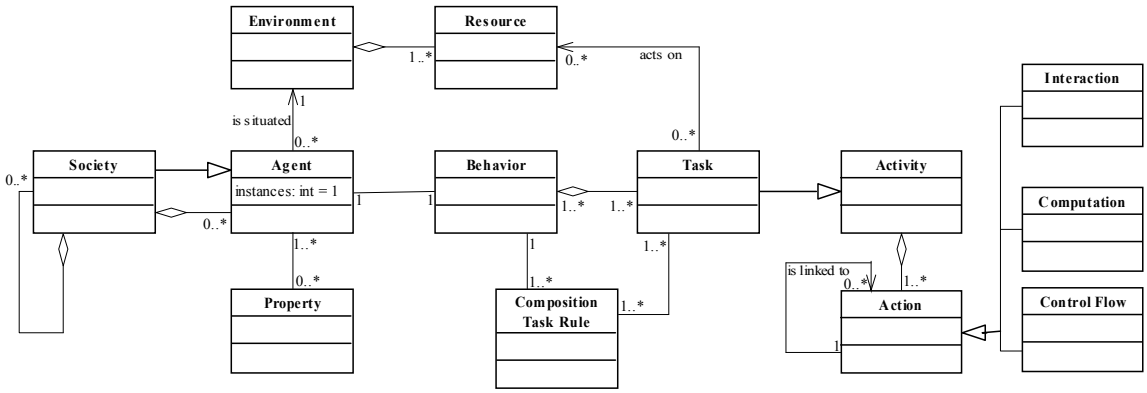


Figure 2: The CIM metamodel.

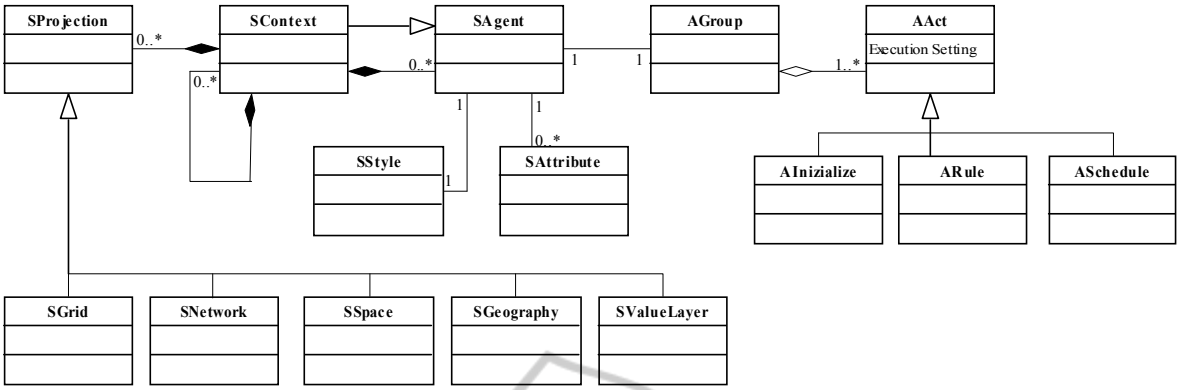


Figure 3: The PIM metamodel.

2.3 From CIM to PIM

With reference to the proposed MDA-based process (see Figure 1) a target model can be obtained by transforming a source model on the basis of a mapping between the respective metamodels which is provided in terms of both mapping rules among corresponding concepts and additional guidelines which enable to obtain instances of concepts of the target metamodel from instances of concepts of the source metamodel (OMG, 2003). This section deals with the mapping between the CIM and the AMF-based PIM metamodels, proposed in Section 2.1 and 2.2 respectively, and, in particular, concerns with the definition of a mapping (see Section 2.3.2) enabling to transform the entities of a CIM model into PIM entities by taking into account specific aspects of the AMF-based PIM metamodel (Karow, 2006) (see Section 2.3.1). The subsequent generation of several PSMs (and code for the related ABMS platforms) from the obtained PIM can be then easily carried out by the visual and Eclipse-based modeling environment provided by the AMF framework (AMP, 2011).

2.3.1 Main Aspects of an AMF-based PIM

Some main aspects have to be considered in the definition of an AMF-based PIM; in this section, the focus is on those which are relevant since they affect the simulation execution of the derived PSMs and which, in particular, concern the proper definition of the *Execution Setting* of an AAct, and the exploitation of *SAttributes* to enable communication among SAgents (see Figure 3).

An AMF-based PIM is defined according to a time-stepped driven simulation approach (the simulation time is incremented in fixed steps) (North, 2007), in which, at each simulation step *t*, a set of AAct instances which can be executed and their execution order are defined. Specifically, in a step *t:* (i) for each AAct, belonging to the *AGroup* of an *SAgent* SA, the number of its instances depends on the number of SA instances; (ii) the AAct *Execution Settings* determine the AAct instances to be executed and their execution order. In particular, the Execution Setting of an AAct is characterized by the tuple <*startingTime, period, priority*> where: (i) *startingTime* is the first simulation step at which the instances of the AAct are to be executed; (ii) for each instance of the AAct, *period* is the number of simulation steps which must elapse between two subsequent executions; (iii) in a simulation step the *priority* value affects the execution order of the *enabled* AActs instances (an AAct is *enabled* at the *simulation step t* if *t* is equal to the AAct *startingTime* which is incremented by a multiple of its *period*).

As in a simulation step *t* all enabled AAct instances (regardless of whether they belong to a specific SAgent instance) belong to the same set, *Enabled(t)*, from which the AActs are scheduled for execution on the basis of their *priority* (see Figure 4), the AAct Execution Settings have to be properly defined to guarantee right execution order between AAct instances of not only the same SAgent instance (intra-agent AAct interleaving) but also different SAgent instances (inter-agent AAct interleaving). Moreover, as for AActs of the *AInitialize* and *ARule* types *starting Time* and *period* are both fixed to 0 and 1 respectively whereas no fixed settings are associated to *ASchedule* AActs, in defining the AAct Execution Settings, the different AAct types should be also considered (see Figure 3).

With respect to the communication among SAgents, since the *SAttributes* of an SAgent can be freely accessed by all the instances of the SAgent,

and the SAttributes of an SContext by all the instances of all the SAgents in the SContext, communication among instances of the same SAgent (intra-agent communication) can exploit SAgent *SAttributes* whereas communication among instances of different SAgents (inter-agent communication) can be enabled by SContext *SAttributes*.

Finally, the design of SAgent communications should take into account how random choices among the enabled AAct (see Figure 4) affect the values of the *SAttributes* on which the communication is based.

2.3.2 Mapping between the CIM and PIM Metamodels

CIM to PIM model transformations require the definition of *mapping rules* among concepts of the source and target metamodels along with additional *guidelines* for managing transformations which, due to the different abstraction level between the concepts of the reference metamodels, cannot be completely automated (Karow, 2006).

The transformation of a CIM into a PIM starts by transforming each *Society* into a Simulation Context (*SContext*) and any enclosed Society into a (sub)SContext of the corresponding enclosing Society. *SAttributes* of each SContext are, then, originated by the *Properties* of the corresponding Society. The next step consists in transforming each *Agent* belonging to a Society into an *SAgent* of the corresponding SContext, generating the SAgent SAttributes on the basis of the Agent Properties, and introducing the SAgent *AGroup* which groups the *AActs* constituting its behavior.

On the basis of the set of *Resources*, which compose the Environment in which Agents are situated, a set of *SProjections*, whose types (SNetwork, SGrid, SSpace, SGeography, SValueLayer) depend on the characteristics of the mapped Resources, are then introduced in the corresponding SContext.

At this point, AActs associated to each SAgent are to be defined on the basis of the behavior of the corresponding Agent which is composed by a set of *Tasks* organized according to *Composition Task Rules*. This transformation is not direct as requires to take into account the specific aspects of both an AMF-based PIM (see Section 2.3.1) and the simulation scenarios to be represented. In particular, due to different communication mechanisms provided by CIM and PIM metamodels, the former based on incoming and outgoing signals (see Section 2.1) and the latter on shared SAttributes (see Section 2.3.1), Tasks which involve *Actions* of the interaction type can be grouped into a single AAct whose *Execution Setting* is defined on the basis of the Composition Task Rules associate to the grouped Tasks (see Section 3.2 for an example). Moreover, both AAct Execution Settings and related AAct types (*AInizialize*, *ARule*, *ASchedule*) must be set not only to ensure compliance with the Composition Rules of the corresponding Tasks but also to guarantee intra and inter-agent AAct interleavings (see Section 2.3.1) which adhere to the simulation scenarios under consideration (see Section 3.2). Finally, *Actions* which constituted the Tasks mapped into an AAct have to be properly realized by exploiting the wide set of predefined functions provide by AMF (AMP, 2011).

Although the transformation of a CIM to PIM cannot be completely carried out automatically as it requires to deal with the above discussed issues, a QVT/R-based representation (OMG, 2010) (Taentzer, 2005) of the discussed mapping has been also obtained to enable an automatic generation of a basic PIM structure to be manually refined. As an example, in Figure 5 the rule for transforming an Agent into an SAgent is reported by using the QVT/R graphical notation (OMG, 2008).

```
ActScheduling (t) {
  AAI = Enabled(t); /* Enabled(t) returns the set of enabled AAct instances at t */
  while (not empty AAI) {
      MPE = maxPriorityEnabled(AAI) ; /* maxPriorityEnabled(AAI) returns a set
                                         consisting of the AAct instances with maximum priority in AAI */
      AAI = AAI - MPE;
      while (not empty MPE) {
        aa = randomGet(MPE); /* randomGet(MPE) returns an AAct instance randomly
                                chosen in (and removed from) MPE */
      execute (aa);
      }
  }
}
```

Figure 4: Execution of an AMF-based simulation step.

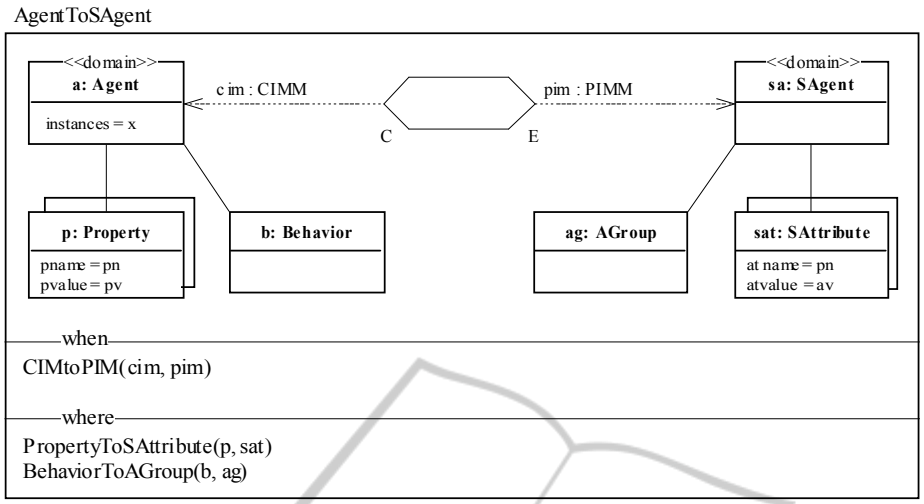


Figure 5: A Mapping Rule expressed by using the QVT/R graphical notation.

3 EXPLOITING THE PROPOSED MDA-BASED PROCESS

In this section, the MDA-based process for ABMS proposed in the previous sections is exemplified with reference to the well-known *Demographic Prisoner's Dilemma* which was introduced by Epstein in 1998 (Dorofeenko, 2002) and is able to represent several social and economic complex scenarios in which interesting issues regard the identification of starting configurations and conditions that allow initial populations to reach stable configurations (in terms of both density and geographic distribution). Specifically, in these scenarios k players are spatially distributed over an n-dimensional toroidal grid. Each player is able to move to empty cells in its von Neumann neighborhood of range 1 (*feasible* cells), is characterized by a fixed pure *strategy* (c for cooperate or d for defect) and is endowed with a level of wealth w which will be decremented or incremented depending of the payoff earned by the player in each round of the *Prisoner's Dilemma* game played during its life against its neighbors (Dorofeenko, 2002). The player dies when its wealth level w becomes negative, whereas, when w exceeds a threshold level w_b, an *offspring* can be produced with wealth level w_0 deducted from the parent and plays using the same strategy as the parent unless a mutation (with a given rate m) occurs. A player also dies if its *age* exceeds a value age_{max} which was randomly fixed when the player was created.

3.1 The CIM Model

For the *Demographic Prisoner's Dilemma*, the CIM model envisages a *DPDGame* Society of k *Player* Agents which are situated in an Environment which includes a *Grid* Resource constituted by an n-dimensional toroidal grid. Main Properties of the *DPDGame* Society are Prisoner's Dilemma *payoffs*, initial and threshold wealth levels (w_0, w_b), and mutation rate (m), and those of the *Player* Agent are its wealth level w, *age*, and *strategy*. The Behavior of the *Player* Agent is obtained by composing the set of Tasks reported in Table 1 according to the Composition Task Rules shown in Table 2; corresponding UML Activity diagrams are reported in Figure 6.

3.2 The PIM Model

In this section, the transformation from the defined CIM to a PIM is detailed with reference to a simulation scenario where all players are required to play exactly one round in a simulation step.

The transformation from the CIM to a PIM is enabled by the mapping between the CIM and PIM metamodels defined in Section 2.3.2 which originate: the *DPDGame* SContex from the DPDGame Society, the *Player* SAgent from the Player Agent, the *GameSpace* SProjection from the *Grid* Resource, the *Acts* (with their related *Execution Settings*) associated to the *Player* SAgent from the Tasks and associated Composition Task Rules composing the Behavior of the Player.

In Table 3 the Acts derived for the *Player* SAgent along with the associated Tasks (see Table 1 and 2) and Execution Settings are reported. As the

Table 1: Identified Tasks.

Task Id	Task Name	Description
T1	Walk	The player can move to a *feasible* cell of the *Grid*.
T2	Challenge	*If the von Neumann neighborhood (of range 1) of the player is not empty* the player communicates its strategy to its randomly selected opponent player.
T3	Update Age	The player age is incremented by 1.
T4	Fission	*If the player's wealth level w is greater than the threshold* w_b a new child player can be created in a *feasible* cell of its parent and endowed with w_0 and the same strategy of the parent (unless a mutation with rate *m* occurs). The wealth level of the parent player is decremented by w_0.
T5	Die	*If the wealth level of the player is negative or its age is greater than* age_{max} the player is removed from the *Grid*.
T6	Accept Dare	*When the strategy of an opponent player is provided* the player strategy is communicated to the opponent and the earned payoff is added to the player's wealth level.
T7	Update Wealth Level	*If the strategy of an opponent player is provided* the earned payoff is added to the player's wealth level

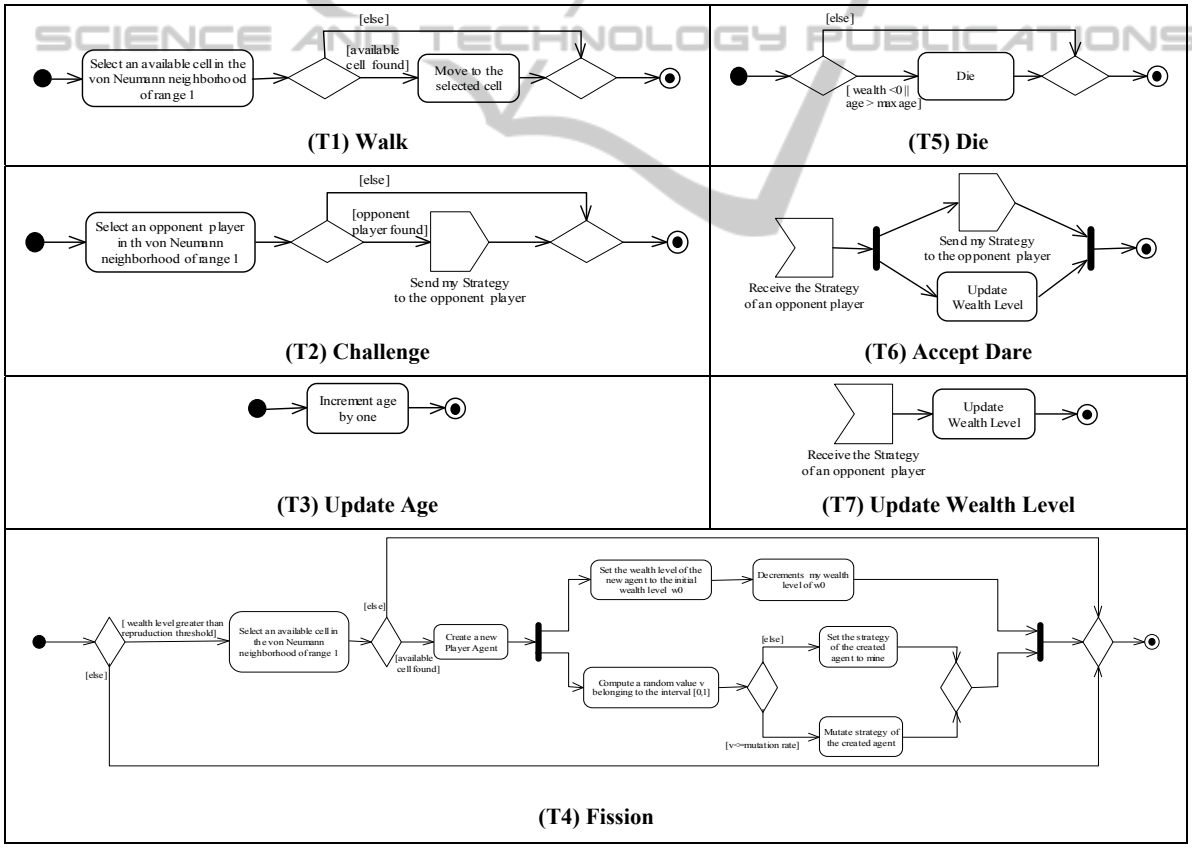


Figure 6: The UML activity diagrams of the Player Agent tasks.

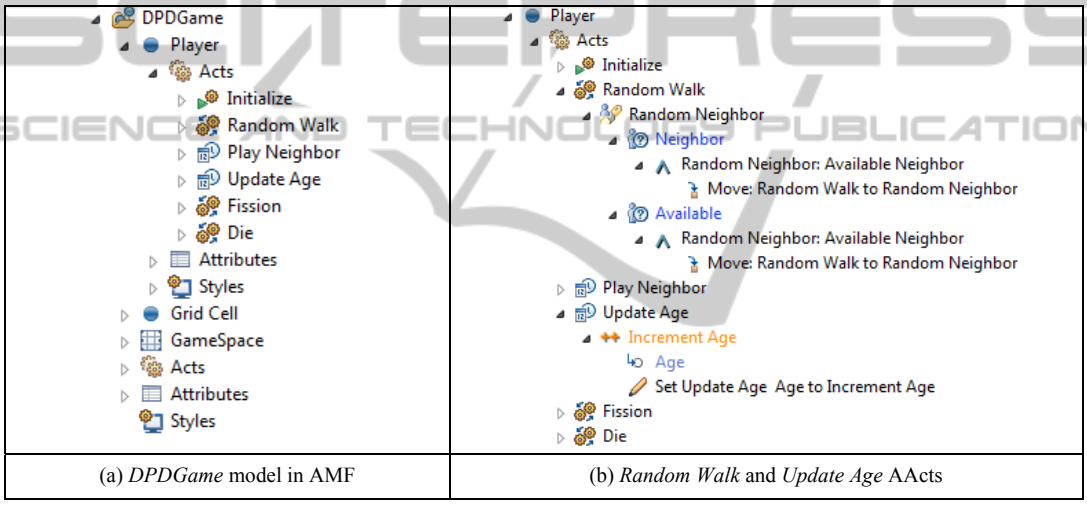
AMF communication mechanism among instances of an SAgent is based on access to the SAttributes of the SAgent (see Section 2.3.1), a single AAct (*Play Neighbor*) is derived from tasks T2, T6 and T7 which carried out this kind of communication. Execution Settings of the AActs in Table 3 are characterized by both *startingTime* and *period* equal to one to guarantee that all the Player SAgents

Table 2: Composition Task Rules.

Task Id	Set of Enabling Tasks
T1	-
T2	{T1}
T3	{T1}
T4	{T7}
T5	{T3, T4}
T6	{T2}
T7	{T6}

Table 3: Group of Acts (AGroup) for the Player Agent.

AAct	AAct Execution Setting	Tasks
Random Walk	<1,1, a>	T1
Play Neighbor	<1,1, b>, with b<a	T2, T6, T7
Update Age	<1,1, c>, with c<a	T3
Fission	<1,1, d>, with d<c & d<b	T4
Die	<1,1, e>, with e<d	T5



(a) *DPDGame* model in AMF

(b) *Random Walk* and *Update Age* AActs

Figure 7: The AMF-based PIM model of the DPDGame.

perform all their AActs in each simulation step, and *priorities* are set on the basis of the Compositions Task Rules (see Table 2).

On the basis of the type of AActs in Table 3, which can be either ARule or ASchedule (see Section 2.3.1), several PIM definitions can be obtained. In Figure 7.a an example of a PIM model representation, obtained by exploiting the visual and Eclipse-based modelling environment provided by AMF, is reported in which *Random Walk*, *Fission* and *Die* AActs are set to the ARule type, whereas *Play Neighbor* and *Update Age* AActs are set to the ASchedule type. Moreover, an AAct of the AInizialize type (*Inizialize*) has been introduced for setting up the *SAttributes* of the *DPDGame* SContext and the *Player* SAgent. In Figure 7.b the definition of the *Random Walk* and *Update Age* AActs is reported where the actions associated to each AAct are defined by exploiting the wide set of functions provided by AMF.

Starting from this definition of the PIM model, AMF is able to automatically generate the PSM models and the related code for the ABMS platforms which are currently supported: Repast Simphon (North, 2005), Ascape (Parker, 2001) and Escape (AMP, 2011). The simulation of the system can then be executed in a target simulation environment andsimulation results can be thoroughly analyzed by exploiting several analysis tools (as Matlab, R, VisAd, iReport, Jung) which can be directly invoked from the environment.

4 CONCLUSIONS

It is widely agreed that significant benefits can be

obtained by exploiting the Agent-Based Modeling and Simulation (ABMS) approach in several application domains (e.g. financial, economic, social, logistic, physical science, chemical, engineering). However, to promote widespread adoption of the ABMS, well-defined processes, modeling techniques, and tools which are able to fully support domain-experts (with often limited programming expertise) are currently lacking.

In this context, the paper has aimed to extend the benefits of the emerging software engineering Model-Driven approach in the ABMS context by proposing an OMG MDA-based process which enables ABMS practitioners to obtain platform-independent agent-based models which are easy to verify (as they conform to meta-models), modify and update (as they are mainly constituted by *visual* diagrams based on the UML notation) with significantly reduced programming and implementation efforts (as the simulation code is automatically generated from the defined models). Although there are interesting proposals which exploit Model-Driven approaches for developing agent-based conceptual and simulation models, the proposal differs from all others as it fully complies with MDA so providing an effective platform independent approach for ABMS.

Starting from the visual modeling environment offered by the AMP Eclipse Project, which has provided the reference framework (AMF) for the definition of *Platform-Independent* and *Platform-Specific simulation Models* as well as code generation, an integrated ABMS environment can be obtained for supporting both the conceptual modeling of the system (in terms of *Computation Independent Models*) and model transformations enabled by the defined meta-models and related mapping. Ongoing research efforts are underway devoted to the definition and extensive experimentation of a full-fledged ABMS methodology which, by exploiting the MDA-based approach, is able to seamlessly guide domain experts from the analysis of a complex system to its agent-based modeling and simulation.

REFERENCES

The AMP project, http://www.eclipse.org/amp/.

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

Bernon, C., Cossentino, M., Gleizes, M. P., Turci, and P., Zambonelli, F., 2004. A Study of some Multi-agent Meta-Models. *Agent Oriented Software Engineering V, revised selected papers*. LNCS, Volume 3382, Springer.

Dorofeenko, V., Shorish, J., 2002. Dynamical Modeling of the Demographic Prisoner's Di-lemma. *Computing in Economics and Finance, Society for Computational Economics.*

Garro, A., Russo, W., 2009. Exploiting the easyABMS methodology in the logistics domain. *Proceedings of the Int'l Workshop on Multi-Agent Systems and Simulation (MAS&S'09) as part of the Multi-Agent Logics, Languages, and Organisations Federated Workshops* (MALLOW 2009), Turin, Italy, September 7-11, 2009.

A. Garro, W. Russo. easyABMS: a domain-expert oriented methodology for Agent Based Modeling and Simulation. *Simulation Modelling Practice and Theory*, Vol. 18, pp. 1453-1467, 2010, Elsevier B.V., Amsterdam, The Netherlands.

Hahn, C., Madrigal-Mora, C., and Fischer, K., 2007. Interoperability through a Platform-Independent Model for Agents. *Enterprise Interoperability II, New Challenges and Approaches*. Springer London.

Henderson-Sellers B., Giorgini P. (editors), 2005. *Agent-oriented methodologies*. Idea Group Publishing, Hershey, PA.

Iba, T., Matsuzawa, Y. and Aoyama, N., 2004. 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.

Karow, M., Gehlert, A., 2006. On the Transition from Computation Independent to Platform Independent Models. *In Proc. of the 12th Americas Conference on Information Systems*, Acapulco, Mexico, August 2006.

Nebrijo Duarte, J.,de Lara, J., 2009. ODiM: A Model-Driven Approach To Agent-Based Simulation. *In proc. of the 23rd European Conference on Modelling and Simulation,* Madrid, Spain, June 9th - 12th, 2009.

North, M. J., Howe, T.R., Collier, N.T. and Vos, J.R., 2005. Repast Simphony Runtime System. *In Proc. of the Agent 2005 Conference on Generative Social Processes, Models, and Mechanisms,* Chicago, IL.

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

Object Management Group (OMG). *Model Driven Architecture (MDA) Guide Version 1.0.1, 2003*. Available at http://www.omg.org/cgi-bin/doc?omg/03-06-01.

Object Management Group (OMG). *Meta Object Facility (MOF) Specifications (version 2.0, 2006)*. Available at http://www.omg.org/spec/MOF/2.0/.

Object Management Group (OMG). *MOF Query/Views/Transformations (QVT) Specifications (version 1.0, 2008)*. Available at http://www.omg.org/spec/QVT/1.0/.

Object Management Group (OMG). *Unified Modeling Language (UML) Specifications (version 2.2, 2009).*

Available at http://www.omg.org/spec/UML/2.2/.

Object Management Group (OMG). *Model Driven Architecture (MDA) Specifications, 2010*. Available at http://www.omg.org/mda/specs.htm.

Parker, M. T., 2001. What is Ascape and Why Should You Care?. *J. Artificial Societies and Social Simulation* 4(1).

Schauerhuber, A., Wimmer, M., and Kapsammer, E., 2006. Bridging existing Web modeling languages to model-driven engineering: a metamodel for WebML. *In proc. of the sixth international conference on Web engineering* (ICWE'06). Palo Alto, CA, ACM press.

Taentzer, G., Ehrig, K., Guerra, E., Lara (de), J., Lengyel, L., Levendovszky, T., Prange, U., Varró, D., and Varró-Gyapay, S., 2005. Model Transformation by Graph Transformation: A Comparative Study. *In Proc. of the ACM/IEEE 8th International Conference on Model Driven Engineering Languages and Systems*, Montego Bay, Jamaica, 2005.