

# A Model-Driven Approach to Close the Gap between Business Requirements and Agent-Based Execution

Christian Hahn<sup>1</sup>, Dmytro Panfilenko<sup>2</sup>, Klaus Fischer<sup>1</sup>

<sup>1</sup> Agents and Simulated Reality (ASR)  
at the German Research Center for Artificial Intelligence (DFKI),  
Campus D3 2 Stuhlsatzenhausweg 3,  
66123 Saarbrücken, Germany  
Christian.Hahn@dfki.de

<sup>2</sup> Institute for Information Systems (IWi)  
at the German Research Center for Artificial Intelligence (DFKI),  
Campus D3 2 Stuhlsatzenhausweg 3,  
66123 Saarbrücken, Germany  
Dima.Panfilenko@iwi.dfki.de

**Abstract:** The current state-of-the-art in developing software applications is to design the systems based on visual design tools and take the resulting design artifact as a base to manually code the software application. The process of transferring the (business) requirements to executable code involves several different parties which makes the whole process again very error prone. In this paper, we present a model-driven approach to overcome the gap between business requirements on the one side and multiagent systems on the other, as we consider the use of agents for implementation beneficial in contrast to more traditional approaches like the WS-BPEL engine.

## 1 Introduction

Service-oriented architectures (SOAs) have emerged as a direct consequence of specific business and technology drivers that have materialized over the past decade. From the business side, major trends such as the outsourcing of non-core operations and the importance of business process re-engineering have been key influences driving the surfacing of SOA as an important architectural approach to business information technology (IT) today. From the SOA side, adequate mechanisms need to be explored to combine business requirements and the underlying execution engines. Therefore, often the principles of model-driven development (MDD) [3] and metamodeling are applied.

The aim of this paper is to present a model-driven architecture for managing and implementing interoperable business processes through SOAs and multiagent systems (MASs). We aim at filling the gap between business requirements made on a strategic level and the execution models on the implementation level. Though, it is possible to

execute business models with traditional means, e.g. communicating workflow engines for E-business protocols like RoseetaNet, the use of MASs for implementation seems to offer various advantages.

The model-driven approach to close the gap between business requirements and executable agent systems has been developed in the SHAPE<sup>1</sup> (Semantically-enabled Heterogeneous Service Architecture and Platforms Engineering, <http://www.shape-project.eu/>) project. SHAPE provides an integrated development environment that brings together MDD with the SOA paradigm and integrates other technologies like MAS, Semantic Web, Grid, and P2P. The technology developed in the project are centered around SoaML [8], a metamodel for describing service-oriented landscapes that is standardized in the Object Management Group (OMG), which is extended with metamodels for the other technology platforms and advanced service engineering techniques

As a consequence of the development in the area of the service modeling the Model Driven Architecture (MDA) increases the level of abstraction of this and related concepts to a new state. MDA's goal is the faster system development through the model transformations from one level into another. These models are classified by the MDA concept into three levels of abstraction, namely the Computation Independent Model (CIM) level, the Platform Independent Model (PIM) level and the Platform Specific Model (PSM) level ([4], [5]).

As the most of the existing MDA-approaches [6] focus on PIM- and PSM-level and the model-to-model transformation between them, the more conceptual CIM-level is often as assumed to be present and is not investigated further. Hence, a real life software system development project comes to a problem during development in the starting phases where the conceptual modeling on the CIM level and even more unstructured verbal or media information about the application's domain is in play. As there is an existing standard for modeling services on the PIM level – SoaML – that is connected to the agent modeling on the PSM level, we introduce the link between the CIM level modeling with the aid of CIMFlex and services modeling on the PIM level with SoaML through conceptual mapping rules for the model-to-model transformation. CIMFlex combines the expressiveness of Business Process Modeling Notation (BPMN) and the ARIS notation.

The remainder of this paper is structured as follows: In Section 2, the model-driven service engineering approach is presented used in our approach. Afterwards, Section 3 illustrates the main concepts of the Service-Oriented Architecture Modeling Language (SoaML), followed by Section 4 detailing the mappings between the business level and SoaML. Section 5 then presents the agent-modeling approach called PIM4Agents and defines the mappings between SoaML and PIM4Agents. Section 6 represents related work, followed by Section 7 naming future extensions. Section 8 then concludes this paper.

---

<sup>1</sup> SHAPE is a European Research Project under the 7<sup>th</sup> Framework Program, detailed information can be found at <http://www.shape-project.eu/>

## 2 Model-Driven Service Engineering

A central part of the SHAPE technology is model transformations that define the basis for (semi-)automated transformation among several model types, and in particular enable the MDE-based approach for integrated top-down modelling from the CIM level to the PSM level.

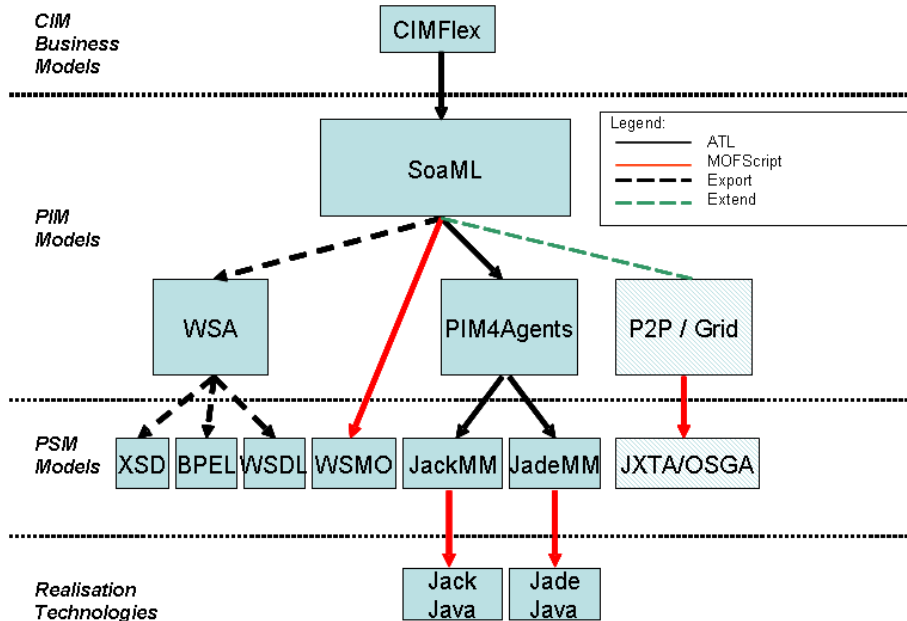


Figure 1: SHAPE Model Transformation Architecture

The model transformation architecture (cf. Figure 1) of SHAPE illustrates the core language used within the project, their relationship to the abstraction levels CIM, PIM and PSM as well as their relationship to other languages defines through model transformations, either model-to-model or model-to-text.

On the highest level, business models encompass business rules, processes, services and other issues such as contracts involving humans and organizations to achieve business goals. These conform to the metamodel of CIMFlex, which supports the user to create and refine a semi-formal model of a business process, an organisational structure, a data structure or business rules based on the input coming from the domain users. The editor is able to create, change and store these types of models in EPC or BPMN notation.

The middle layer contains the results of the proposal as transformation engines, extended SOA models, the standardized SoaML and extensions for semantically-enabled heterogeneous architectures (ShAML). This architecture allows the realization of one of the main goals of SHAPE namely to provide a transformation engine that

maps business models to SOA models which are then transferred to the various execution platforms.

The model transformation architecture will in particular support the following model transformations:

#### **CIM to PIM transformation**

- Model transformation between CIMFlex and SoaML: The challenge in transforming CIMFlex models to SoaML is to generate the appropriate system relevant constructs for SoaML according to the generic business context on CIM level. CIMFlex supports in its initial version the model-to-model transformation by making use of the Atlas Transformation Language (ATL) [9].

#### **PIM to PIM transformation**

- Model transformation between SoaML and Web Services: The transformation between SoaML and Web Services is done through a model-to-model transformation. The transformation will produce three kind of models form a single SoaML models. It will produce XML Schemas for information description, WSDL files for interface description and finally BPEL files for behavioural specification.
- Model transformation between SoaML and PIM4Agents: The transformation between SoaML and PIM4Agents is done through a model-to-model transformation using ATL.

#### **PIM to PSM transformation**

- Model transformations between PIM4Agents and the metamodel of JACK [12] and JADE [13], which are both agent execution platforms. The model transformations between the PIM4Agents metamodel and the metamodels of JACK/JADE (JackMM/JadeMM) are specified through a model-to-model transformation using ATL.
- Model transformation between SoaML and WSMO: The model transformation between the metamodel of SoaML and WSMO is specified through a model-to-text transformation.

In the remainder of this paper, we focus on the model transformation path from CIMFlex to SoaML and from SoaML to PIM4Agents.

### 3 Service-Oriented Architecture Modeling Language

The Service-Oriented Architecture Modeling Language (SoaML) [8] is standardized in OMG. It describes a UML profile and metamodel for designing services.

The goals of SoaML are to support the activities of service modelling and design and to fit into an overall model-driven development approach. The SoaML profile supports the range of modelling requirements for service-oriented architectures, including the specification of systems of services, the specification of individual service interfaces, and the specification of service implementations.

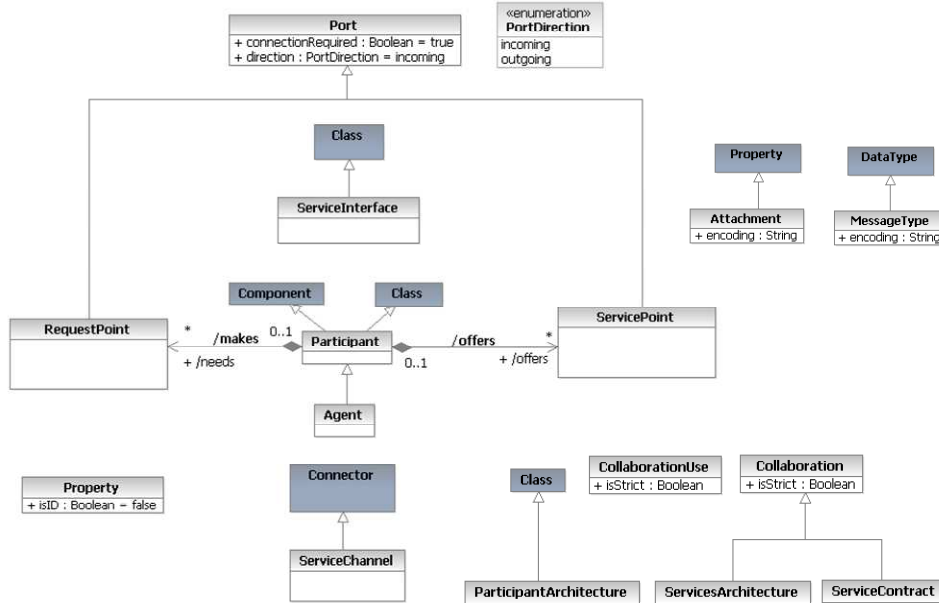


Figure 2: The SoaML profile

The SoaML profile (see Figure 2) extends the UML2 metamodel to support an explicit service modelling in distributed environments. This extension aims to support different service modelling scenarios such as single service description, service-oriented architecture modelling, or service contract definition. The main extension areas are:

- **Participants** to define the service providers and consumers in a system. A Participant may play the role of service provider, consumer or both. When a participant acts as a provider it contains ServicePoints, and when a participant acts as a consumer it contains RequestPoints.
- **Service interfaces** to describe the operation provided and required to complete the functionality of a service. A ServiceInterface can be used as the protocol for a ServicePoint or a RequestPoint.

- **Service contracts** to describe interaction patterns between service entities. A `ServicesContract` is used to model an agreement between two or more parties. Each service role in a `ServiceContract` has a `ServiceInterface` type that usually represents a provider or consumer.
- **Service data** to describe service messages and message attachments. The `MessageType` is used to specify the information exchanged between services, attached to rather than contained in the message.
- **Services and participant architectures** to define how a set of participants works together for some purpose by providing and using services. A `ServicesArchitecture` or a `ParticipantArchitecture` describes how participants work together by providing and using services expressed as `ServiceContracts`.

## 4 From CIMFlex to SoaML

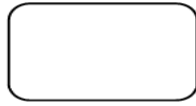

In this section we provide some aspects of the high CIM-level service modeling with the aid of the BPMN. This notation is well-known and established since the beginning of the 21<sup>st</sup> century, moreover it has been standardized and there are more than 50 products, both commercial and open-source, providing the implementation of this standard [11]. The particular considerations with respect to modelling services by the business users are that there is a little awareness of the services by CIM-level users, on the one hand, and even if there would be any knowledge about it, there are no direct constructs describing the services on the CIM-level in the BPMN notation anyway. Of course, the upcoming BPMN 2.0 standard includes the services modeling and the according constructs for it, but it only rules out the second, more technical problem, and not the first one – understanding.

For solving this problem, we propose a semi-automated approach in this section based on a model-to-model transformation from CIM-level BPMN models to PIM-level SoaML-based models. Those models on the higher abstraction level in BPMN would be analyzed through a set of mapping rules and would result in a service model representing according constructs and architectures needed for the comprehensive PIM-level model as a basis for the further transformation to the PSM-level. The concrete mappings are as follows:

*Task to ServiceInterface* – as a task describes an activity that is possibly providing a useful output that could be consumed by the participants of the process, it can be then assigned to Service Interface construct in this mapping, as it gives the abstract interface for the job done and at the same time doesn't give further specification of the workflow implementing this task.

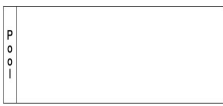
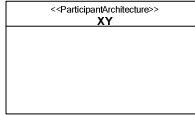
Table 1 Task to ServiceInterface

BPMN Description	Symbol	SoaML Description	Symbol
------------------	--------	-------------------	--------

Task		ServiceInterface	
------	---	------------------	---

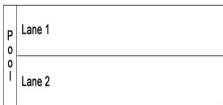
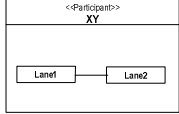
*Pool to Participant / ServicesArchitecture* – a pool in BPMN stands for a business entity or a participant of a process, on the one hand. It also can be structured with respect to further participants of the process, thus creating a participants' hierarchy. These two points together put the pool on a role of a candidate for a Participant or Service Architecture, depending on the modeller's intention.

Table 2 Pool to Participant / Service Architecture

BPMN Description	Symbol	SoaML Description	Symbol
Pool		Participant Architecture (Service Architecture)	

*Lane to Participant* – a lane represents a participant or a department in BPMN and is situated in a pool, thus showing the two-tier hierarchy. In order to show the possibility for further subdivision (which is also ongoing in the current BPMN2 proposals), the lane is first mapped to a Participant and next tiers of this hierarchy are constructed using the role constructs described below.

Table 3 Lane to Participant / Service Architecture

BPMN Description	Symbol	SoaML Description	Symbol
Lane		Participant Architecture (Service Architecture)	

*Lanes to ServiceContract* – this transformation also reflects the service contract from the CIM level model into the SoaML and later into PIM for agent interaction specification. There is also a task sequence connected by a sequence flow, but the participants are represented through different lanes in the same pool, thus showing another possibility for a participant architecture modelling. The two tasks that belong to a service contract also share a data object.

Table 4 Lanes to ServiceContract

BPMN	Symbol	SoaML Description	Symbol
------	--------	-------------------	--------

Description			
Lane1 → Lane2		ServiceContract	

## 5 From SoaML to MAS

In this section, a rough overview on the platform independent metamodel for MASs (PIM4Agents) is given. Afterwards, the model transformations between SoaML and PIM4Agent are discussed.

### 5.1 Platform-Independent Metamodel for Multiagent Systems

For modelling MAS and in particular the protocol and the common data model, we developed a platform independent modeling language for MAS called domain-specific language for multiagent systems. The abstract syntax of this language is defined by a platform independent metamodel for MAS called PIM4Agents

The PIM4Agents metamodel [14] that defines the abstract syntax of the modelling language for MASs is a visual platform-independent model that specifies MASs in a technology independent manner. It represents an integrated view on agents in which different components can be deployed on different execution platforms. The PIM4Agents metamodel defines modelling concepts that can be used to model six different aspects or views of an agent system that are listed below:

- **Agent view** describes single autonomous entities, the capabilities they have to solve tasks and the roles they play within the MAS. An Agent has access to a set of Resources from its surrounding Environment. These Resources may include information or ontologies the Agent has access to. Furthermore, the Agent can perform particular DomainRoles that define in which specific context the Agent is acting and Behaviours that define how particular tasks are achieved.
- **Organization view** describes how autonomous entities cooperate within the MAS and how complex organizational structures can be defined. The Organization is a special kind of Agent and can therefore perform DomainRoles and have Capabilities which can be performed by its members. In addition to the Agent properties, an Organization may have its own internal Protocols specifying (i) how the Organization communicates with other Agents be them atomic Agents or complex Organizations and (ii) how organizational members are coordinated.
- **Role view** covers feasible specializations of the Role concept (i.e. DomainRoles used to partition the organizational space and Actors used to



define the message exchange within Protocols) and how they could be related to each other.

- **Interaction view** describes how the interaction between autonomous entities or organizations takes place. Each interaction specification includes the Actors involved and in which order ACLMessages are exchanged between these Actors in a protocol-like manner.
- **Behavioural view** describes how Plans are composed by complex control structures and simple atomic tasks like sending or receiving a Message and how information flows between those constructs. A Plan specifies the agents' internal processes.
- **Environment view** contains any kind of Resource (i.e. Service, Object) that is dynamically created, shared, or used by the Agents.
- **Deployment view** describes the run-time AgentInstances that are involved in the system and how these are assigned to the organization's roles.

To close the gap between design and implementation, we provide generic model transformations from PIM4Agents on the platform independent level to two underlying execution platforms (i.e. JACK or JADE on the platform specific level).

## 5.2 Model Transformations: From SoaML to PIM4Agents

*ServicesArchitecture to Organization:* The concept of a ServicesArchitecture nicely corresponds to the concept of an Organization PIM4Agents as both refer to roles that interact in accordance to some predefined processes. However, and this is the main differences between both constructs, a ServicesArchitecture does not perform any role to the outside. Hence, the generated Organization is more or less utilizes as a social structure providing the space for interaction. Hence, the Organization does neither own any Plans nor perform any DomainRole to the outside and hence should not be considered as an autonomous entity in the MAS, but rather as a form of grouping the necessary autonomous entities to fulfill the service. Likewise, the resulting Organization does not own any kind of knowledge, capability, or resource.

*ParticipantArchitecture to Organization:* A ParticipantArchitecture illustrates, in contrast to a ServicesArchitecture, a concrete entity in the system described. Thus, the target Organization may perform a DomainRole which is either required inside ServicesArchitectures/ServiceContracts or even in other ParticipantArchitectures. Moreover, the Organization may own certain knowledge which is used by the source ParticipantArchitecture.

*ServiceContract to Interaction:* The main purpose of a ServiceContract is to define the roles that agreed on the contract and how these interact with each other which is expressed through any kind of UML behavior. Hence, for representing a ServiceContract in PIM4Agents, the right choice is an Interaction, which defines how the exchange of messages between Actors is specified.

*ServiceInterface to Collaboration:* A ServiceInterface defines a bi-directional service which includes the two roles provider and requester as well as the choreography which specifies the global interaction. In PIM4Agents, the concept of Collaboration is the best match, as it binds AgentInstances to (i) DomainRoles of the Organization and (ii) Actors of the Interaction.

*Participant to AgentInstance:* In the same manner as ParticipantArchitutures are transformed to Agents, a Participant is mapped to an AgentInstance. The agent type of the AgentInstance is deduced from the ParticipantArchitecture that specifies the Participant.

*UML Behavior to Plan:* Any kind of UML Behavior is transformed to a Plan in PIM4Agents. This Plan then specifies the internal processes of the Organization.

*Interface to Capability:* A UML Interface defines a collection of operations and/or attributes that ideally define a set of processes. In order to represent this in an adequate manner in PIM4Agents, the concept of a Capability depicts the perfect match, as both, operations as well as attributes can be included into one of its Plans.

## **6 Related Work**

Only few works exist aiming to bridge the gap between business-oriented approaches and MASs. Taveter [15] presented an agent-based approach for business modeling where Agent-Object Relationship Modeling Language (AORML) [16] is used as underlying agent modeling language. However, normally, agent languages are not the preferred paradigm of business analysts when it comes to designing the particular business requirements. Particular tailored languages are normally used for this purpose. Consequently, Endert et al. [17] presented a transformation between BPMN and JIAC IV (Java-based Intelligent Agent Componentware) [18] to bridge the gap between business process languages on the one hand and agent-based systems on the other hand. However, beside the fact that only a single platform is involved in their model transformation architecture, the even more problematic issue is that in most cases the gap between business languages like BPMN and agent platform cannot be automatically bridged. An intermediate level like defined by SOAs is often considered as more beneficial.

## **7 Future Work**

The future work on connecting the high-level service modeling with the systems based on services comprises the alignment of the current BPMN to SoAML mapping with the upcoming new version of the BPMN. As there should be service constructs directly in the BPMN CIM-level methodology, it would be much easier to put this

high-level notation with more technical description of a system on a PIM-level with the aid of SoaML in one line together in order to prepare it for the transformation to the PSM-level eventually resulting in an initial snap of the working system's code.

Another point of the future work concerning the high-level service modeling will be the study of the acceptance and evaluation of the comprehensiveness grade of the new version of the upcoming BPMN standard. That is, the question arises how much of the information can be put on the CIM-level that is through different transformations "pushed down" towards the system code. The real problem is not that the information can or cannot be modeled at the highest level, but the question of the understandability of the modeling constructs by the business level users.

A third area of future work is the direct combination of BPMN 2.0 with agent-based systems defined by PIM4Agents. The question that have to be answered in this respect is how much information has to be manually added on the PIM4Agents level after applying BPMN to PIM4Agents model transformation to generate executable agent code.

## 8 Conclusions

This paper represents a model-driven approach to close the existing gap between business requirements specified using existing business modeling languages and agent technologies. To realize this, we defined two model-to-model transformations: The first transformation is specified between CIMFlex and SoaML, the second transformation maps SoaML models to PIM4Agents models. By utilizing the code generators of PIM4Agents, the generated models can be mapped to executable code based on the agent platforms JACK and JADE.

This approach allows the specification of business requirements using BPMN and the mapping to agent code. On each level of the SHAPE model transformation architecture, details with respect to the underlying language and technology can be added that normally requires different roles to be involved, from the business analyst to the agent programmer.

## 9 References

- [1] Erl, T.: SOA - Entwurfsprinzipien für serviceorientierte Architektur. Addison-Wesley, München et al. 2008
- [2] Mathas, C.: SOA intern. Hanser, München 2008
- [3] Stahl, T.; Völter, M.: Modellgetriebene Softwareentwicklung. Techniken, Engineering, Management. dpunkt, Heidelberg 2005
- [4] Frankel, D. S.: Model Driven Architecture. Applying MDA to Enterprise Computing. Wiley, Indianapolis (2003)

- [5] Object Management Group: MDA Guide Version 1.0.1, [www.omg.org/docs/omg/03-06-01.pdf](http://www.omg.org/docs/omg/03-06-01.pdf)
- [6] Mellor, S.J.; Scott, K.; Uhl, A.; Weise, D.: MDA Distilled: Principles of Model-Driven Architecture. Addison-Wesley, (2004)
- [7] Object Management Group: Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification. <http://www.omg.org/cgi-bin/apps/doc?ptc/05-11-01.pdf>
- [8] Object Management Group: Service oriented architecture Modeling Language (SoaML) – Specification for the UML Profile and Metamodel for Services (UPMS). <http://www.omg.org/docs/ad/08-08-04.pdf>
- [9] INRIA & LINA: ATLAS Transformation Language (ATL) project documentation. <http://www.eclipse.org/gmt/am3/doc/>
- [10] Hahn, Christian (2009-01-14) SHAPE – Deliverable 5.1: Model transformation and deployment architecture description. [http://www.shape-project.eu/wp-content/uploads/2009/01/shape\\_d51.pdf](http://www.shape-project.eu/wp-content/uploads/2009/01/shape_d51.pdf)
- [11] Object Management Group / Business Process Management Initiative: BPMN implementations. [http://www.bpmn.org/BPMN\\_Supporters.htm](http://www.bpmn.org/BPMN_Supporters.htm)
- [12] Papasimeon, M., Heinze, C.: Extending the UML for designing JACK agents. In: Proceedings of the Australian Software Engineering Conference (ASWEC 01). (2001)
- [13] Bellifemine, F., Bergenti, F., Caire, G., Poggi, A.: 5. In: JADE - a Java agent development framework. Volume 15 of Multiagent Systems, Artificial Societies, and Simulated Organizations. Springer-Verlag, Berlin et al. (2005) 125–147
- [14] Hahn, C.: A platform independent agent-based modeling language. In: Proceedings of the Seventh International Conference on Autonomous Agents and Multiagent Systems (AAMAS). (2008) 233–240
- [15] Taveter, K. A. (2004). A Multi-Perspective Methodology for Agent-. Oriented Business Modelling and Simulation, PhD thesis
- [16] Wagner, G. (2003). The Agent-Object-Relationship meta-model: Towards a unified view of state and behavior, Information Systems 28(5): 475–504
- [17] Endert, H., Hirsch, B., Küster, T. and Albayrak, S. (2007). Towards a mapping from BPMN to agents, in J. Huang, R. Kowalczyk, Z. Maamar, D. L. Martin, I. Müller, S. Stoutenburg and K. P. Sycara (eds), Proceedings on the Internal Workshop on Service-Oriented Computing: Agents, Semantics, and Engineering (SOCASE 2007), AAMAS 2007 Honolulu, HI, USA, May 14, 2007, Vol. 4504 of Lecture Notes in Computer Science, Springer Verlag, Berlin et al., pp. 92–106
- [18] Albayrak, S. and Wiecek, D. (1999). Jiac - a toolkit for telecommunication applications, Proceedings of the Third International Workshop on Intelligent Agents for Telecommunication Applications (IATA '99), Springer-Verlag, London, UK, pp. 1–18