Methodologies, Platforms, Ontologies and Techniques for a Multi-Agent Architecture for Simulation of Traffic with Communications

Pedro Fernandes and Urbano Nunes

{pedro, urbano}@isr.uc.pt

Technical Report

October 2007
Methodologies, Platforms, Ontologies and Techniques for a Multi-Agent Architecture for Simulation of Traffic with Communications

Pedro Fernandes, Urbano Nunes
ISR – Institute of Systems and Robotics
Department of Electrical and Computer Engineering, University of Coimbra
Polo II, Coimbra, Portugal
+351 239 796 287
{pedro, urbano}@isr.uc.pt

ABSTRACT
Inter-vehicle communications, in the context of Intelligent Transportation Systems, will probably bring a significant improvement in both traffic safety and efficiency. In order to evaluate in what measure this is true, traffic simulations that take into account the communications between vehicles are needed. Several approaches are being pursued to this end, many of them working on the integration of existing traffic simulators and network simulators. However, this integration is, in most cases, done offline, not allowing real-time operation. Another line of research proposes traffic simulation with the use of agents, allowing the development of more adaptive systems. However, these proposals don't consider implicit vehicle communications and its integrated simulation.

In this paper, we propose a novel agent-based architecture, in which the simulation and management of the inter-vehicle communications are integrated in simulated agent-based vehicles and infrastructure, in a hierarchical multi-agent environment. In this architecture, each simulated vehicle is in itself a multi-agent system, with specialized agents to manage driving maneuvers, vehicle data, localization, communication filtering and interface. Inter-vehicle communications are managed by an agent-based module that simulates real wireless communications between vehicles, using the appropriate standards. To allow interoperability, the platform supporting the development of the proposed multi-agent system complies with FIPA specifications.

Moreover, since the approach to a multi-agent system involve multiple choices about the appropriated methodologies, platforms, ontologies, communication standards, among others, a resumed roadmap is also presented.

Keywords

1 INTRODUCTION
The number of persons residing in urban spaces is increasing every day. The people rely mostly on individual vehicles, congesting the transportation networks. Studies and simulations of traffic have been made for decades, through macroscopic, mesoscopic and microscopic traffic simulators [1]. Recently, in the context of Intelligent Transportation Systems (ITS), vehicle communications are being developed, aiming above all the improvement of traffic safety and efficiency. To study such systems, efforts to integrate traffic simulators and network simulators have been pursued [2-4].

Concerning the use of agents in traffic simulation, several approaches have been made [5-8]. It is obvious that any agent-based system must provide communication functionalities between agents. However, to simulate real wireless communications between vehicles, the use of explicit communication simulation has to be considered. Yet, very few proposals of such systems consider its use explicitly.

The integration in a system of both traffic and network simulations may be considered a complex task, due to a wide set of reasons, such as the intrinsic complexity traffic theory, the wireless network transmissions, the real-time constraints and the distributed nature of the system among others.

At the present, traffic theory does not account to driver behavior changes due to the newly existence of communications. Therefore, equation-based modeling is not the most appropriate method to use in simulation, as agent-based modeling may provide a more appropriate emulation of driver behavior [9]. Moreover, the agent-based modeling allows the development of a mode adaptive system, and although system validation may be more difficult, it can be done at both system and individual levels. Due to the distributed nature of traffic, we believe that agent-based technology is more appropriate than equation-based technology.

The communication system we propose will have both a distributed component and a locally centralized one.

2 PROBLEM DESCRIPTION
The model proposed consists of a novel multi-agent system that manages the communications inside each vehicle and between each of them and the infrastructure. The simulation of the communication between vehicles will be managed by a common component shared by vehicles in the neighborhood. The architecture will be tested in the context of an intersection, where the management of communications and localization of the vehicles will have both a distributed and a centralized component. This option aims to provide simulation functionalities at communication level that, in the reality, would be provided by the transmission media. Moreover, localization of hazardous situations (vehicles without communications, pedestrians) is better provided by centralized facilities.
The architecture proposed to the multi-agent system is depicted in Figure 1.

2.1 Multi-agent architecture
The main agents involved in the proposed architecture are briefly described in the following paragraphs.

2.1.1 Network Simulator
The main function of the Network Simulator (NS) is to receive all communications between Communication Manager agents, and simulate the network transmission between them, considering the environment and the location of each one. Appropriate communication standards must be used by this agent, namely Dedicated Short Range Communication (DSRC), ruled by Wireless Access in Vehicle Environments (WAVE), defined in IEEE 802.11p and IEEE 1609.x standards. Other standards to consider are Communication Air Interface Long and Medium Range (CALM) [82].

2.1.2 Intersection Traffic Rule Arbiter
The Intersection Traffic Rules Arbiter (ITRA) must deal with intersection control of traffic, recording all traffic events and dealing with resolution of conflicts between User Managers (UM). With low traffic throughput, we may have a distributed control of traffic, where UM may agree with the right of way of each other, always under ITRA supervision. As traffic flow grows, ITRA will have to validate all UM decisions, eventually overcoming some of them. In a high traffic flow scenario, all traffic rules decisions must be taken by ITRA, and vehicles become “data probes” of the centralized traffic rule management.

2.1.3 Communication Manager
This agent (CM) manages communications between the vehicle and external systems, such as the infrastructure and other vehicles. In both cases, NS is used as intermediary, to simulate wireless network transmissions. In the case of vehicle communications, each system communicates through its own CM.

2.1.4 Message Broker
The Message Broker (MB) must manage all internal messages, and has the incumbency of filtering and its prioritization, ensuring that critical messages are dealt first by the appropriate agents. In this scheme, MB may delay low priority messages or, is some cases, even discard such messages.

2.1.5 User Manager
UM main function has to do with decisions about the right of way of the vehicle, with the agreement of all vehicles in direct neighborhood, always under ITRA supervision. As stated in §2.1.2, as traffic flow grows, the decisions are taken by ITRA, in a centralized manner. To avoid deadlocks, all the decisions – distributed or centralized– must be taken with anticipation, allowing the forecast of possible deadlocks and its resolution before they actually occur.

2.1.6 ADInterface
The ADInterface (ADI) agent deals with the selection of the most appropriate message interface to the driver, taken in account the type of message and its urgency. For this end, auditory, visual, or haptic message presentation methods must be timely chosen.

2.1.7 Localization
Localization agent determines the localization of the vehicle in the intersection map, using GPS data and an intersection beam signal, and compares its position with neighbor vehicles positions, periodically transmitted through wireless communications. This agent must decide whether the situation is critical, based on position and vehicle data, and warn UM in case of imminent danger.

2.1.8 Vehicle
Vehicle agent gathers vehicle data (e.g. speed, acceleration, brakes, steer) and feeds Localization agent with that information. UM receives also similar feedback. Moreover, this agent gets commands issued by Driver agent.

2.1.9 Driver
Driver agent deals with the control of the whole vehicle. It receives information, whether critical or not, via ADI agent and responds accordingly to that information and the type of driver modeled. For that purpose, Driver agent maintains a driver type database. This agent issues commands to Vehicle agent directly, and indirectly through ADI.

2.1.10 Traffic Simulation Environment
This component isn’t an agent, but the environment where the multi-agent evolves. One of its main functions is to provide communications between agents, in the platform level, allowing appropriate management of agents’ percepts and actions. Graphical presentation of simulation results will also be directly connected with this architecture component.

3 DEVELOPMENT
One of the main difficulties when starting a work based on agents consist on the variety of definitions and concepts of what an agent is, the lack of a unified formal framework for multi-agent models – although some work in progress [10] – and the nonexistence of a generally accepted methodology for the development of agent-based systems.

According to Wooldridge [11], “an agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives”. The autonomy and the situatedness are two important characteristics of agents. Moreover, when agents are situated in
dynamic environments they must be reactive, producing actions in the environment, to respond to changes, observed through percepts. Another characteristic relevant in agents is proactivity, i.e., the pursuit of goals. To do so, agents maintain a belief base – its incomplete knowledge about the environment – and a plan base, which is a base of methods to reach its goals.

When considering multi-agent architecture, issues like organization, coordination, security are also quite relevant.

To develop the system, a disciplined approach is followed, through the overall process. The software engineering process RUP (Rational Unified Process) [12] is taken into account when applicable, relating it to the appropriate agent methodology. Subsequently, an appropriate platform should be chosen [13], along with communication standards between agents – preferably based on open standards – and appropriate ontologies. The simulation platform must also be selected or developed.

The Model-Driven Architecture (MDA) [14], proposed by OMG [15], is based on the separation of the system functionality specification from implementation specification of that functionality on a specific platform, through the development of a computation independent viewpoint (CIM), a platform independent model (PIM), modeled in UML [16], which may be transformed, later, in one or more platform specific models (PSM). The CIM, also called domain model, doesn't show the details of structure of the system. [17] suggests that is precisely in this model, not addressed directly by MDA, that agent-oriented modeling may provide an appropriate set of CIM and PIM concepts transformable one into another.

Due to the several existing methodologies, methods have been proposed to the construction of agent-oriented methodologies, using method fragment integration. In [18] two approaches are considered: a meta-model driven and a development process-driven. The former providing more flexibility in the definition of MAS methodologies and meta-models, but difficult to integrate fragments, while the later providing more simplicity in the construction of methodologies, but more rigid and less flexible.

### 3.1 Methodologies

Several proposed methodologies to develop the system may be considered, allowing process development in an organized manner. Prometheus [19], Gaia [20], Tropos [21], along with Nikraz et al. proposal [22], are just some of the wide existing examples in the literature. However, not all existing methodologies are appropriate to every problem. Some of them aim generality (e.g., Gaia), loosing in detail. Other ones focus more on specific platforms and languages, losing generality but gaining detail and adaptability.

In [23] Sturm and Shehory propose a framework for evaluating agent-oriented methodologies. Defined the evaluation parameters – concepts and properties, notations and modeling techniques, process and pragmatics – the authors proceed with Gaia evaluation, as an example.

In [24], Cernuzzi et al. distinguishes between software processes and methodologies, relating the later with how to perform a task in specific phases of the process, while the former is associated with basic questions like who, when and how much. The authors uses the four process models they identify in the literature as most relevant – (1) waterfall like, (2) evolutionary and incremental, (3) transformation and (4) spiral – to classify the existing agent-oriented methodologies, according to the phases of software development process. Gaia and Prometheus, two of the more general methodological approaches are classified in the waterfall-like group, although the later is admittedly considered as being easily adaptable to evolutionary or spiral process models. Evolutionary and incremental methodologies are more appropriated for agent-based systems where emergent behavior is a requirement. However, the Prometheus and Gaia methodologies allow more control over agent behavior.

The Mensa Project [25] focused on the state of the art of agent-oriented software engineering (AOSE) methodologies and multi-agent systems (MAS) infrastructures. The authors acknowledge the existence of a technical and conceptual gap between agent methodologies and infrastructures as the source of difficulty when transforming agent-oriented methodologies products into implementable system over MAS infrastructure.

In the following paragraphs a resume of the main characteristics of some methodologies are presented.

#### 3.1.1 Prometheus

This methodology was proposed by Lin Padgham and Michael Winikoff in 2002 [26]. According to the authors, the reason why they proposed a new methodology was the methodology claimed detail, support of BDI (Beliefs, Desires and Intentions) [27] agents, scaling ability and tool support. Interestingly, the authors don't consider Prometheus as a waterfall-like process model, as Cernuzzi et al. do; for the contrary, not following such model was a required criteria for the development of Prometheus in the first place.

The main three phases of Prometheus methodology are: (1) system specification, (2) architectural design and (3) detailed design. In the first one system goals and the basic functionalities of the system are identified, use case scenarios illustrating the system’s operation are developed and the interface between the system and its environment in terms of actions and percepts are specified. In the second phase decision about what types of agents will be implemented and its descriptors are made. The static system structure is captured with system overview diagram, and the dynamic behavior of the system is captured with interaction diagrams and interaction protocols. During the last phase the capabilities of agents are refined, producing the agent overview diagram and capability descriptors.

![Figure 2: System Overview](image-url)
To support design and development of multi-agent systems using Prometheus methodology, Winikoff and Padgham developed the Prometheus Design Tool (PDT) [28], that implements the three phases of Prometheus and process some consistency checking.

The platform usually addressed by PDT is JACK Intelligent Agents™ [45], although other tools allow the use of different platforms, namely Jadex [46].

In our view, this methodology, not followed in a strict manner but in an iterative way, is appropriate to the development of the initial framework proposed.

An example of the system overview proposed, using Prometheus methodology is depicted in Figure 2.

3.1.2 Gaia

Gaia methodology aim generality, to allow its use for a broad type of agent-based systems. However, this characteristic, which is one of its strengths, is also its most pointed weakness, since the detailed design phase and implementation have intentionally been left out. The main phases of the methodology are: (1) system requirements, (2) analysis phase, which includes both the interactions model and the roles model, (3) design phase that includes agent model, services model and acquaintances model.

As stated above, this methodology does not provide a detailed design phase or an implementation one. However, several proposals to extend Gaia have been made. Moraitis et al. [29] propose an enhancement of a previous roadmap to extend Gaia with detailed design and implementation phases, using JADE frameworks. In [30], an extension of Gaia with the incorporation of an agent design phase and an enhancement of the methodology with the use of iterations is proposed, since the methodology was considered incomplete [31]. In [32], supportive extensions are proposed, consisting in Gaia-based practical design and modeling tools to better support real-world applications. In [33], is proposed an extension of Gaia with capabilities to facilitate the development of complex open systems.

Table 1. Methodologies classification [26]

<table>
<thead>
<tr>
<th>Methodology</th>
<th>Process Model</th>
<th>Phases</th>
<th>Requirements</th>
<th>Analysis</th>
<th>Design</th>
<th>Implementation</th>
<th>Verification</th>
<th>Deployment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaia</td>
<td>Waterfall like</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Roadmap</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prometheus</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MaSE</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AOR</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evolutionary and Incremental</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPM/MAS</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MASSIVE</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ingenias</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tropos</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PASSI &amp; Agile PASSI</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Platform categorization.

<table>
<thead>
<tr>
<th>Platform Categorization</th>
<th>Social and economical coordination (L5)</th>
<th>Agent behavior (L4)</th>
<th>Ontology-based domain model (L3)</th>
<th>Agent-specific infrastructure services (L2)</th>
<th>System environment base services (L1)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[partially]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3.1.3 Methodology to use with JADE

Nikraz et al. presented a methodology [24] that, using JADE platform [42], allows the analysis and design of multi-agent systems. This methodology seems quite appropriate to the intended use. Although the planning phase is not formally addressed, analysis phase includes use cases, initial agent type identification, responsibilities identification, acquaintances identification, agent refinement and agent deployment information. The design phase addresses agent splitting/merging/renaming, interaction specification, ad-hoc interaction protocol definition, message templates, description to be registered/searched (Yellow Pages), agent-resource interactions, agent-user interactions, internal agent behaviors, definition of an ontology and content language selection. The implementation and testing phases are not formally addressed.

3.1.4 Other methodologies

ROADMAP [33], Tropos [20], SODA [52], MESSAGE [53], MaSE [34], MAS-CommonKADS [35], AOR [36], OPM/MAS [37], MASSIVE [38], Ingenias [39], DESIRE [40], PASSI [78] and AgilePASSI [41], are some other methodologies that we find in literature. In Table 1, a classification of some of the methodologies, presented in [26], is reproduced.

3.1.5 Methodology selection

In this work, we think Prometheus is an appropriate methodology for initial system development. All the relevant phases are covered conveniently, and PDT tool allows consistency and completeness checking through the steps of each of the phases.

3.2 Platforms

Choosing the right platform for the problem domain at hand is not a trivial task. The choice is closely connected with the methodology adopted. [13] proposes a categorization of agent platforms with respect to reference architecture, with five layers, L1 to L5, as presented in Table 2. In the group of middleware platforms, the authors include JADE [42], ADK [79], FIPA-OS [80] and DIET [43]. These platforms are conform FIPA standards [44], allowing interoperability among platforms. In the reasoning platforms, based on specific agent architectures, such as reactive, proactive and hybrid, are included JACK [45], Jadex [46], Jason [47] and SOAR [48] platforms. The social platforms include MadKit [49] STEAM [50] and JACK SimpleTeams [51].
### 3.2.1 Jade

Jade (Java Agents DEvelopment framework) [54] is probably the most used agent-oriented middleware. It is an open source distributed middleware system, compliant with FIPA [44] specifications, that implements both white and yellow pages, agent mobility, ontologies and content languages, among other features, in a flexible platform allowing easy extension with add-on modules [55]. JADE does not provide, however, direct support to the development of BDI agent architectures. However, Jadex, briefly presented in the next paragraph, implements such architecture over JADE.

### 3.2.2 Jadex

Jadex [46, 57] is a software framework for the development of goal-oriented agents following the BDI model. Since JADE platform does not allow direct implementation of this model, Jadex, using JADE as the underlying middleware platform, allows the creation of rational agents. It has as one of its objectives the adoption of software engineering perspective for agent description, where beliefs are represented in an object-oriented fashion. This differs from both Jason, where beliefs are represented as first-order logic predicates, and JACK, that uses relational models. Jadex agents have two main components: an agent definition file (ADF), coded in XML, and Java code. ADF specifies the beliefs, goals, plans and their initial values. Java code defines the procedural part of plans, which accesses BDI facilities through APIs. Jadex BDI metamodel is specified in XML Schema.

### 3.2.3 Jason

Jason [47, 58] is an interpreter of the an extended version [59] of AgentSpeak(L) [60], allowing agents to be distributed over the network using Simple Agent Communication Infrastructure (SACI) [61], although this mobility is not yet provided within the platform in the present version. Jason is available as open source and uses jEdit [62] as IDE.

### 3.2.4 Jack

JACK® [45, 63] is a commercial agent platform from Agent-Software [64], which uses syntactic and semantic extensions of Java that allows the implementation of BDI agents. These extensions solve other Java limitations in the development of agents, due to its fixed semantics. The platform allows also the execution of agents with message and name server services and several development tools, such as a design tool, a graphical plan editor and tools for debugging purposes.

### 3.2.5 Platform selection

Although the platform usually used by Prometheus user is JACK, we consider the use of open source platform as an added value, as any need to tweak the source code is possible. Moreover, the compliance with FIPA specifications is very important to grant interoperability of the systems. That’s why we plan to use JADE platform for the multi-agent system development, on top of which, Jadex may be used to allow reasoning agent implementation, despite some limitations intrinsic of Java language used.

Another option is to use a rule engine, such as Jess [65], that allows agents to make decisions, based on declarative rules. In [66], integration of JADE and Jess is exemplified.

### 3.3 Ontologies

Communication is a valuable tool for agents to interact, exchange information and request services. As stated by Sycara and Paolucci [67], “Ontologies provide agents the basic representation that allows them to reason about interactions but also, and most importantly, ontologies provide agents with shared knowledge that they can use to communicate and work together”. At the present, Ontology Web Language (OWL) [68], is the language of the Semantic Web that is being standardized by the World Wide Web Consortium. The Ontology Web Language for Services (OWL-S) [69] aims to provide the service descriptions' rich semantics on the Web, complementing SOAP, WDSL and BPEL4WS.

Any multi-agent system has to deal with ontological issues. A agent platform must allow the use of content language (e.g. FIPA- SL Content Language Specification) [70], in addition to communication languages (e.g. FIPA-ACL Agent Communication Language) [71], as JADE does.

### 3.4 Simulation

Multi-Agent Based Simulation (MABS) is considered the support of choice for the simulation of complex systems [72], replacing other micro-simulation techniques, most of them object-oriented. However, when developing such systems, even if agents are present in the domain and design levels, they seem to be less present in the operational level, where object-oriented or procedural languages are frequently used, resulting in agent implementations with the lack of some of the characteristics agents should have, e.g. autonomy, proactiveness and adaptiveness.

[73] compares multi-agent modeling to standard modeling and presents some guidelines for multi-agent based selection based on problem characteristics.

### 4 RELATED WORK

The use of intelligent agents in traffic simulation is an emergent area of research [5-8]. Many proposals exist in literature. [74] describes a model of a reactive agent that is used to control a simulated vehicle. [75] presents a multi-agent simulation models with applications to Berlin traffic. [76] proposes an agent-based hierarchical architecture to develop centralized and decentralized platoons. In [77] an urban traffic control system using multi-agent technology is presented. The system architecture is structured in three layers, using a hierarchical architecture.

We have not found in the literature attempts to model and simulate communications between vehicles using a multi-agent system, in an integrated way. Many proposals [2-4] look for the interconnection of existing network simulators, (e.g. ns2 - The Network Simulator, or SWANS - Scalable Wireless Ad hoc Network Simulator) with traffic simulation platforms. These works uses most of the times equation-based traffic microsimulation platforms (e.g. SUMO - Simulation of Urban Mobility) or simulators based on cellular automata, which presents characteristics more close to mesoscopic simulators [81]. However, this approach of interconnect traffic and network simulators has the drawback of make difficult to keep up with real-time constraints.

### 5 CONCLUSIONS

We propose, in this technical report, some concepts and approaches of an architecture in which the simulation and manage-
ment of the inter-vehicle communications are integrated in simulated agent-based vehicles and infrastructure, in a hierarchical multi-agent environment. For the development of work in progress to succeed, we presented a short survey of existing meta-models, methodologies and platforms, and suggested some possible choices, matching methodologies and platforms to allow appropriate system implementations.

6 FUTURE WORK
This work is in the stage of the definition of the model, its architecture, and specifications of the multi-agent system. Future work will take place in the subsequent stages, such as the analysis and design (general and detailed) of the system using appropriate methodology, the implementation of the solution in the selected platform, the validation process and final deployment.

7 ACKNOWLEDGMENTS
This work was supported by Institute of Systems and Robotics and Fundação para a Ciência e Tecnologia under contract NCT04/POSC/EEA/SRI/58016/2004.

8 REFERENCES


[41] Chella, A., Cossentino, M., Sabatucci, L. and Seidita, V. From PASSI to Agile PASSI: Tailoring a design process to meet new needs. In 2004 IEEE/WIC/ACM International Joint Conference on Intelligent Agent Technology (IAT-04), Beijing, China, Sept 2004


[52] SODA: Societies in Open and Distributed Agent spaces. DOI= http://soda.alice.unibo.it


[64] Agent Software. DOI= http://www.agent-software.com/


