A Multi-agent System Architecture for Requirements Management in the Extended Enterprise

A. Smirnov & A. Krizhanovsky
St.Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences, St.Petersburg, Russia

R. Roy & C. Kerr
Cranfield University, Cranfield, United Kingdom

ABSTRACT: The objective of this paper is to present the architecture of a multi-agent system (MAS) for the effective management of product requirements and constraints in an Extended Enterprise. Analysis and design of the MAS is based on the Gaia methodology which defines a comprehensive set of agent roles, agent types, services and protocols. The goal of the agent platforms is to simplify development while ensuring standard compliance (e.g. for interoperability between different agent systems). A comparison of the modern platforms (Grasshopper, JADE, and ZEUS) for agent creation is presented.

1 INTRODUCTION

The concept of an agent as a self-contained problem-solving system capable of autonomous, reactive, proactive, social behaviour represents yet another tool for the software engineer (Wooldridge & Jennings 1998). Agent technology is appropriate for applications that can be naturally modelled as societies of interacting autonomous entities.

The use of agents can be useful in several fields of application (Gasser 2001; Puliafito et al. 2000): information retrieval, information fusion, electronic commerce, distributed management, distributed computation and collaborative applications.

Although the lack of design tools might prevent the multi-agent architect from exploiting its benefits (Bergenti & Poggi 2000), agent based systems are used actively in industrial and commercial fields of applications: manufacturing, electronic commerce and business process management (Luck et al. 2003; Roy et al. 2003).

There are a number of characteristics in the Extended Enterprise (EE) domain that make it a suitable application area for MAS's (Camarinha-Matos & Afsarmanesh 2001). The most important examples of such characteristics include:

- EE's are composed of distributed, heterogeneous and autonomous components which is a situation easily mapped into MAS.
- The coordination and distributed problems of the EE.
- Decision-making with incomplete information.
- The effective execution of distributed business processes requires quick reactions from enterprise members. Each company in an EE can be presented by an agent.
- The phase of EE formation in which it is necessary to select partners and distribute tasks, shows market characteristics and negotiation needs that have been research issues in MAS's.
- The flexible modelling paradigm of the EE (replacement of partners, changes in partners' roles, etc.).
- Modularity of infrastructure.
- Autonomy and cooperative behaviour.

The main objective of this paper is to present the architecture of an electronic requirements management (e-RM) system based on the Gaia methodology. The aim of Gaia design is to transform the analysis models into a sufficiently low level of abstraction than traditional design techniques. Gaia is concerned with (i) how a society of agents cooperate to realize the system-level functions and (ii) what is required of each individual agents in order to do this (Wooldridge et al. 2000).

The aim of the e-RM project is to improve the business process capability of the OEM to manage requirements and constraints for mechanical designs with their suppliers using a digital process. The study will primarily focus within the automotive industry. The research team involves Nissan Technology Centre - Europe (Nissan) (as an OEM), Johnson Controls Automotive (UK) Ltd. (JCA) (as a Supplier), SDRC (as a software vendor), and SMMT (as Automotive Trade Association). The Cranfield team will contribute in developing an ontology based representation for the requirements and constraints, a knowledge based cost impact analysis method and a generic framework for the information management.

The remainder of this article is structured as follows. The paper introduces to requirement manage-

ment and presents the MAS architecture of the e-RM system. Then the modern platforms for agents' development are presented and compared.

2 MANAGEMENT OF PRODUCT REQUIREMENTS AND CONSTRAINTS

One of the main objectives of requirements engineering (RE) is to improve systems modelling and analysis capabilities so that organizations can better understand critical system aspects before they actually build the system. A topic of increased interest in RE is the analysis and management of the dependencies of among requirements. This topic is called Requirement Interaction Management (RIM) (Robinson et al. 2003).

RIM has five main directions: representations; activities; views of the activity and products; goals of stakeholders; theoretic basis for the representation activities and views.

The representation and activities of RIM are of interest for e-RM:

- *Representations of requirements, interactions, resolutions, and other products.* Researchers are defining the set of terms, or ontology, that describe requirements and their interactions. An ontology is a formal description of entities and their properties, relationships, constraints, behaviors. It provides a common terminology that captures key distinctions and is generic across many domains (Gruninger et al. 2000).
- *Activities for discovery, management, and disposition of interactions.* Researchers are defining the set of techniques, some automated, that manages or aid in managing interactions.

3 MAS DESIGN METHODOLOGIES

There are several software development techniques now that are specifically tailored to MAS. RIO methodology defines the following meta-model concepts: role, interaction, organization and agent. The design phase of RIO consists of two steps: roles' assigning to agents and design of the agent's internal architecture (Gruer et al. 2002). Another approach uses UML-notation to design MAS at the agent level (Bergenti & Poggi 2000).

Gaia methodology makes a clear distinction between the analysis phase, in which the roles and interaction models are elaborated, and the design phase, in which agent services and acquaintance models are developed. Gaia approach tries to avoid any premature commitment, either architectural, or as to the detailed design and implementation process which will follow (Wooldridge et al. 2000).

Gaia is appropriate for the development of large-scale real-world applications. e-RM project meets the requirements of Gaia methodology: (i) inter-agent relationships do not change at run-time, (ii) the abilities of agents and the services they provide are static, in that they do not change at run-time, (iii) the overall system contains a comparatively small number of different agent types.

4 ANALYSIS AND DESIGN OF AN MAS BASED ON THE GAIA METHODOLOGY

The Foundation for Intelligent Physical Agents (FIPA) is a set of basic technologies that can be integrated by developers to make complex systems with a high degree of interoperability. The FIPA model was selected as a reference model in this project. The problem oriented agents are designed in this section.

4.1 *Requirement Statement*

The purpose of the designed system is an effective management of the design requirements and constraints of a product within an Extended Enterprise. The requirements and constraints are presented by means of a developed ontology.

The main function of the system is to provide a tool that allows users (design engineers) to work in a distributed team.

The system has the following requirements:

- Design support. The management of the requirements and constraints for product design is based on expert knowledge. The system has to present the ability to develop and manage the requirements and constraints.
- Cost impact analysis. A cost impact analysis requires a reasoning support in order to manage the great number of dependent constraints.
- Distribution support. Designers will be working in different places on the same model.
- Semantic consistency. Different organization in Extended Enterprise can use different notation, terms, definitions to describe the same concept, while global constraints have to bind them.
- Multilingual support. It is possible that designers speak different languages.

4.2 *Analysis Stage: Roles Model*

The following roles are presented in the system:

- OntologyTransfer requests, gets, and sends an ontology.
- OntoMerger merges ontologies.
- OntoMonitor monitors changes of ontology.
- CostImpactAnalyzer performs cost impact analysis.
- OntoManager checks and maintains ontology consistency.
- OntoInstanceEditor provides a user interface in order to edit (add, change, delete) ontology instances and send/request ontology.

– CostImpactRequestor provides a user interface in order to analyze cost impact.

The following set of tables describes the agents' roles in the system. The role's template (Table 1) used in the Gaia methodology simplifies the role definition. The right-hand column of Table 1 explains the meaning of each row in Tables 2-8.

To describe liveness expression in detail, consider the responsibility of the OntologySender role (Table 2):

OntologySender = (LoadOntology. CheckConsistency. [SendOntology]) $^{\omega}$

This expression says that OntologySender role consists of executing `LoadOntology` activity (activity names are underlined), followed by the `CheckConsistency` activity and the optional protocol `SendOntology`. The sign ω shows that the sequential execution of these protocols and activities is then repeated infinitely often.

Table 1. Template for role schemata

Role	*Name of role*
Description	*short English description*
Protocols and Activities	*protocols and activities in which the role plays a part*
Permissions	*"rights" associated with the role (the type and the amount of resources that can be exploited when carrying out the role.*
Responsibilities *(functionality)*	
Liveness	*liveness responsibilities (the resources that can legitimately be used to carry out the role)*
Safety	*safety responsibilities (the resource limits within which the role executor must operate (what can't be spent))*

Table 2. Role to request/send/get an ontology

Role	*OntologyTransfer*
Description	*This role involves 1) requesting, 2) sending (ensuring that ontology is consistent), and 3) getting an ontology and saving*
Protocols and Activities	*LoadOntology, CheckConsistency, SendOntology, GetOntology, SaveOntology, RequestGetOntology, GetTimestamp, CompareTimestamp, ConfirmRequest.*
Permissions:	change *ontology* connects *addressee* sends *ontologyResult* reads *timestamp // ontology modification*
Responsibilities	
Liveness	*OntologyRecipient = (GetOntology. SaveOntology)* $^{\omega}$ *OntologySender = (LoadOntology. CheckConsistency. [SendOntology])* $^{\omega}$ *OntologyRequestor = (RequestGetOntology. GetTimestamp. CompareTimestamp. ConfirmRequest)* $^{\omega}$ *Resources: local ontology, recipient address, address of ontology owner*
Safety	*If (SendOntology AND ontology is non-consistent) Þ ontologyResult =nil* *recipient address is valid* *waiting (connection) time < T_{max}*
	disk space is enough to save ontology

Table 3. Role to merge ontologies

Role	*OntoMerger*
Description	*This role involves merging ontologies*
Protocols and Activities	*LoadOntology, MergeOntology, GenerateReport, SaveOntology, NotifyMerger, AwaitRequest*
Permissions	reads supplied OntologyTransfer *ontology* writes *resultOntology* writes *report*
Responsibilities	
Liveness	*OntoMerger = (LoadOntology. MergeOntology. SaveOntology. GenerateReport. NotifyMerger, AwaitRequest)* $^{\omega}$ *Resources: local ontologies*
Safety	*Ontology backup version has been saved*

Table 4. Role to monitor changes of ontology

Role	*OntoMonitor*
Description	*This role involves monitoring changes of ontology*
Protocols and Activities	*SendUpdateRequest, AwaitOntoChanges*
Permissions	sends *updateRequest* to OntoInstanceEditor & CostImpactAnalyzer
Responsibilities	
Liveness	*OntoMonitor = (SendUpdateRequest. AwaitOntoChanges)* $^{\omega}$
Safety	*true*

Table 5. Role to analyze cost impact

Role	*CostImpactAnalyzer*
Description	*This role involves cost impact analyzing*
Protocols and Activities	*LoadOntology, LoadRules, AnalyzeCostImpact, GenerateReport, SendReport*
Permissions	reads supplied OntoMerger *ontology* reads *rule* generates *report*
Responsibilities	
Liveness	*CostImpactAnalyzer = (LoadOntology. LoadRules. AnalyzeCostImpact. GenerateReport. SendReport)* *Resources: inference engine, ontology, set of rules*
Safety	*Ontology ≠ nil, Rules ≠ nil, Report ≠ nil, (CostResult = AnalyzeCostImpact, CostResult ≠ nil)*

Table 6. Role to schedule work with ontologies and propagate ontology changes

Role	*OntoManager*
Description	*OntoManager checks and maintains ontology consistency*
Protocols and Activities	AwaitRequest, AwaitRequestPerforming, PushRequestToOrder, PopRequestFromOrder, SendMergeRequest
Permissions	generates *merge of ontologies request* changes *requests order*
Responsibilities	
Liveness	*OntoManager = (AddRequest. || TreatRequest)* $^{\omega}$ *AddRequest = (AwaitRequest.*

	PushRequestToOrder) *TreatRequest = (AwaitRequestPerforming. PopRequestFromOrder, SendMergeRequest)*
Safety	*time ordering of requests treating (FIFO)*

Table 7. Role to work with instances of ontology

Role	*OntoInstanceEditor*
Description	*OntoInstanceEditor provides a user interface in order to edit ontology instances and send/request ontology*
Protocols and Activities	*InstanceAdd, InstanceDelete, InstanceChange, SaveOntology, LoadOntology, AttributeSetValue, RequestGetOntology, GetOntology, RequestSendOntology, SendOntology*
Permissions	changes *ontology instances and their properties* send/request *ontology*
Responsibilities	
Liveness	*OntoInstanceEditor = (Load. Edit. Save)* $^{\omega}$ *Load = ([RequestGetOntology. GetOntology. SaveOntology]. LoadOntology)* *Edit = (InstanceAdd. || InstanceDelete. || InstanceChange. || AttributeSetValue)* $^{\omega}$ *Save = [SaveOntology]. [RequestSendOntology. SendOntology].* *Resources: local ontology*
Safety	*Check validity of instances and attributes values.* *Perform GetOntology if global ontology is newer than local one.*

Table 8. Role to request cost impact

Role	*CostImpactRequestor*
Description	*CostImpactRequestor provides a user interface in order to analyze cost impact*
Protocols and Activities	*SendRequestAnalyzer, AwaitAnswer, GetReport, PrintReport*
Permissions	send *requests // to CostImpactAnalyzer* reads supplied CostImpactAnalyzer *report* prints *report*
Responsibilities	
Liveness	*CostImpactRequestor = (SendRequestAnalyzer. AwaitAnswer. GetReport. PrintReport)* *Resources: UI, ontology*
Safety	*True*

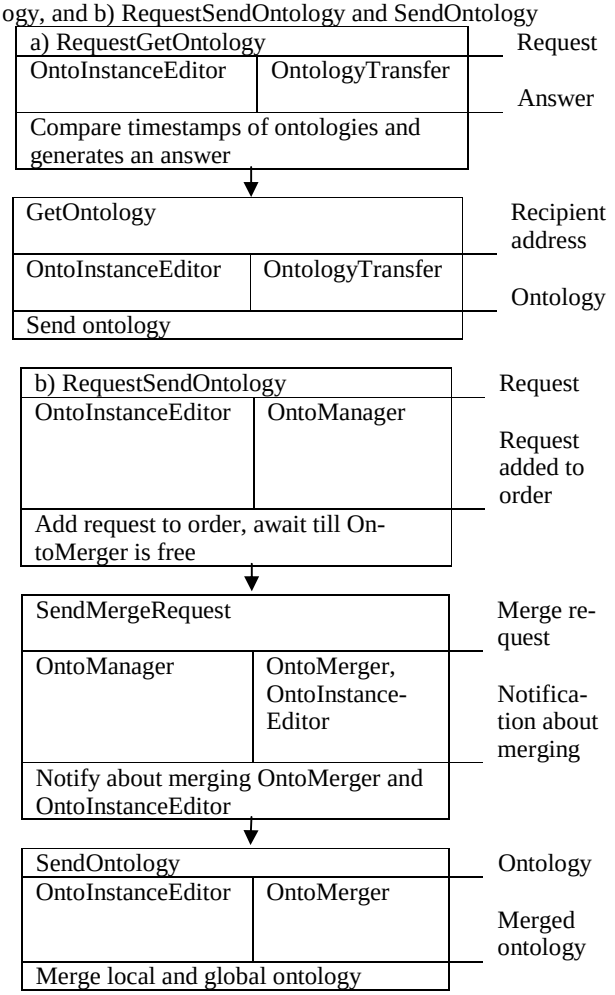
4.3 *Analysis Stage: Interactions Model*

Links between roles are represented in the interaction model. Here a protocol can be viewed as a formally defined and an abstracted pattern of interaction.

The interactions with OntoInstanceEditor role (as a key role) will now be discussed. This role interacts with the OntologyTransfer role in order to obtain an ontology (`RequestGetOntology` and `GetOntology` protocols, Table 9 a). If instances in a global ontology are newer than in local one than `GetOntology` protocol will be performed.

The edition of instances involves the set of activities, such as: `InstanceAdd`, `InstanceDelete`, `InstanceChange`, and `AttributeSetValue`. In order to save result and merge it with global ontology, OntoInstanceEditor interacts with OntoManager and OntoMerger (`RequestSendOntology` and `SendOntology` protocols, Table 9 b). Here OntoManager saves the request and waits until OntoMerger is free, then OntoManager notifies OntoInstanceEditor to send an ontology to OntoMerger and to start merging.

Table 9. Definitions of protocols associated with OntoInstanceEditor role: a) RequestGetOntology and GetOntology, and b) RequestSendOntology and SendOntology



4.4 *Design Stage*

The acquaintance model of agents (Fig. 1) shows the communication pathways between agent types.

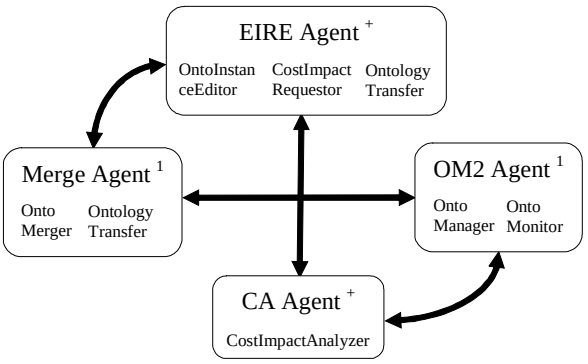


Figure 1. Representation of the acquaintance model

The agent type can be presented as a set of agents roles. An annotation "+" means that there will be one or more instances in the run-time system, and the annotation "1" means that there will be exactly 1 instance of this agent type at run-time.

The acquaintance model defines the communication links that exist between agent types. It does not define what messages are sent, it simply indicates that communication pathways exist.

The service model is the last model at the design stage (list of properties adapted from Wooldridge et al. 2000):

- service is a function of the agent;
- service of one agent is not available to other agents;
- every activity (identified at analysis stage) will correspond to a service, though not every service will correspond to an activity;
- services are derived from the list of protocols, activities, responsibilities and the liveness properties of a role.

The services model for the CostImpactAnalyzer role and CA agent is presented in (Table 10). From the `LoadOntology` and `LoadRules` activities, the *obtain ontology* and *obtain rules* services were derived.

The central service of this agent is *analyze cost impact* associated with the activity `AnalyzeCostImpact`. This service has pre-conditions that an ontology and rules are consistent and a request is valid, i.e. it is applicable to these ontology and rules.

The fourth service is associated with the `GenerateReport` activity. This service takes the output of the *analyze cost impact* service and generates a human-readable report. If the result of cost impact analysis is not null (pre-condition) then the report will be generated (post-condition).

From the `SendReport` protocol, the *send report* activity is derived. There are two pre-conditions for this service that the location of EIRE Agent is valid and the report contains the information about the cost impact.

Table 10. The services model

Service	Inputs	Outputs	Pre-condition	Post-condition
obtain ontology	*Ontology location*	*loaded Ontol.*	*Ontology is available*	*Ontology ≠ nil*
load rules	*Rules*	*loaded Rules*	*Rules are available*	*Rules ≠ nil*
analyze cost impact	*Rules, Ontology and Request*	*CostResult*	*Rules and Ontology are consistent, Request is valid*	*CostResult ≠ nil*
generate report	*CostResult*	*Report*	*CostImpactResult ≠ nil*	*Report ≠ nil*
send report	*EIRE Agent location, Report*		*location is valid Report ≠ nil*	*True*

5 AGENT PLATFORMS

There exists a huge number of approaches, toolkits, and platforms for multi-agent development (Nguyen & Dang 2003; AgentBuilder 2003).

The selection of an MAS platform has to take into account the two points of view: satisfaction of the requirements of an EE (standardization, security, interoperability, etc.), and ease and comfort of the development (documentation, interface, openness, presence of modules for rapid development).

One of the main requirements for enterprise platforms is standardization. The objective of standardization is to learn from the existing experience of the company (Gelete et al 2003). So FIPA compliance is the first requirement for an MAS platform.

Security is domain and platform (implementation) specific — there is no general agent security architecture which is suitable for all applications and implementations. As there are no completed current specifications for agent security within FIPA (Poslad & Calisti 2000), the presence of a security mechanisms provided by the MAS platform is essential.

Several of the most interesting and widely known FIPA-based MAS platforms will now be presented and discussed.

Grasshopper provides new opportunities for the enhancement of electronic commerce applications, dynamic information retrieval, advanced telecommunication services and mobile computing. The region concept facilitates the management of the distributed components, i.e. agencies, places and agents. Each agency is the actual runtime environment for mobile and stationary agents. This consists of two parts, i.e. the core agency (communication, registration, management, transport, security and persistence service) and one or more places (logical grouping of functionality of agents).

JADE (Java Agent DEvelopment Framework) is a software framework for the development of agent applications in compliance with the FIPA specifications for interoperable intelligent multi-agent systems. The goal of JADE is to simplify development while ensuring standard compliance through a comprehensive set of system services and agents (Bellifemine et al. 1999).

The aim of the *ZEUS* project is to facilitate the rapid development of new multi-agent applications by abstracting into a toolkit the common principles and components underlying some existing multi-agent systems. ZEUS provides a set of software components and tools which are used to design, develop and organize agent systems. Moreover, it provides a runtime environment, which enables applications to be observed and other assistant tools such as: reports tool, statistics tool, agents and society viewer etc. ZEUS has an excellent debugging interface, a library of predefined coordination strategies, a general purpose planning and scheduling mechanism, and self executing behaviour scripts. ZEUS

documentation is very weak which can lead to difficulties when creating new applications.

The generalized properties of each discussed platforms is presented in Table 11.

Table 11. Comparing of MAS platforms.

Properties	Grasshopper	JADE	ZEUS
Standards (FIPA, MASIF, CORBA)	(+, +, +)	(+, —, +)	(+, —, —)
Communication (ACL, RMI, KQML)	(+, +, —)	(+, +, —)	(+, —, +)
Good security features	+	+	+
Used in many projects	+	+	+
Good documentation	+	+	—
Good GUI*	+	+	+
Free of charge	+	+	+
Open source	—	+	+

* Graphical User Interface

6 CONCLUSIONS AND FUTURE WORK

The architecture of the multi agent system was developed according to the Gaia methodology. Seven agent roles and four agent types were defined. A set of protocols associated with the OntoInstanceEditor role was presented. At design stage the agent type model, the acquaintance model, and services model were built.

The iterative character of the Gaia methodology helps to regulate the chaos of the design process and helps to 'grasp' the more integral and holistic idea of the system.

There are several issues remaining for future work: (i) design of case studies for validation of the proposed MAS architecture; (ii) development of a software tool for testing the MAS; (iii) analysis of a system complexity for a proposed integrated requirements management system; (iv) development of system architecture to allow localised multi-lingual interfaces.

There are a huge number of modern platforms for agent creation. The most interesting and widely known platforms were presented and compared: Grasshopper, JADE, and ZEUS. The Grasshopper platform was selected for the agents' development.

7 ACKNOWLEDGEMENTS

The authors would like to acknowledge the UK Engineering and Physical Sciences Research Council (EPSRC); Nissan Technical Centre Europe, Johnson Controls Automotive, EDS and the Society of Motor Manufacturers and Traders (SMMT) as the e-RM project sponsors; and the e-RM team members at Cranfield University. This work was also partly supported by the Russian Academy of Sciences (project # 16.2.44 and 1.9), and grant # 02-01-00284 of the Russian Foundation for Basic Research.

8 REFERENCES

AgentBuilder. 2003. A survey of agent construction tools. URL: http://www.agentbuilder.com/AgentTools/index.html

Bellifemine, F., Poggi, A. & Rimassa, G. 1999. JADE — a FIPA-compliant agent framework. *In Proc. of the Fourth International Conference and Exhibition on The Practical Application of Intelligent Agents and Multi-Agents*: 97-108.

Bergenti, F. & Poggi, A. Exploiting UML in the design of multi-agent systems. 2000. In A. Omicini, R. Tolksdorf, and F. Zambonelli, editors, *Engineering Societies in the Agents World*, Lecture Notes in Computer Science 1972: 106–113. Springer. URL: http://lia.deis.unibo.it/confs/ESAW00/pdf/ESAW13.pdf

Camarinha-Matos, L. M. & Afsarmanesh, H. 2001. Virtual Enterprise Modeling and Support Infrastructure Applying Multi-agent System Approach. In Michael Luck et al. (eds), *Multi-Agent Systems and Applications*: 335-364. Springer-Verlag, Berlin.

Gasser, L. 2001. Perspectives on Organizations in Multi-Agent Systems. In Michael Luck et al. (eds) *Multi-Agent Systems and Applications*: 1-16. Springer-Verlag, Berlin.

Gelete, S. Coscuellala, C. & Garcia, J. 2003. Agent assisted concurrent process engineering system for design support. *Concurrent Engineering (CE'2003) - the vision for the Future Generation in Research and Applications*, J. cha, R. Jardim-Goncalvez, A. Steiger-Garcao(eds), A. A. Balkema Publishers, 1: 463-467, Portugal.

Gruer, P., Hilaire, V., Koukam, A., and Cetnarowicz, K. 2002. A formal framework for multi-agent systems analysis and design, Expert Systems with Applications, 23(4):349-355.

Gruninger, M., Atefi, K., & Fox, M.S. 2000. Ontologies to Support Process Integration in Enterprise Engineering. *Computational and Mathematical Organization Theory*, 6(4):381-394.

Luck, M., McBurney, P. & Preist, C. 2003. Agent Technology: Enabling Next Generation Computing (A Roadmap for Agent Based Computing), AgentLink.

Nguyen, T.G. & Dang, T.T. Agent platform evaluation and comparison, 2002. http://pellucid.ui.sav.sk/TR-2002-06.pdf

Poslad, S & Calisti, M. 2000. Towards Improved Trust and Security in FIPA Agent Platforms. *Presented at the Autonomous Agents 2000 Workshop on Deception, Fraud and Trust in Agent Societies*, Spain.

Puliafito, A., Tomarchio, O. & Vita, L. 2000. MAP: Design and Implementation of a Mobile Agents Platform. *Journal of System Architecture*, 46(2):145-162. URL: http://sun195.iit.unict.it/Papers/jsa00.ps.gz

Robinson, W.N., Pawlowski, S.D. & Volkov, V. 2003. Requirements Interaction Management. ACM Computing Surveys, 35(2):132-180.

Roy, R.; Kerr, C. I. V. & Sackett, P. J. 2004. Requirements management for the automotive extended enterprise: The next evolution in digital product development. CIRP 2004 International Design Seminar, Cairo, Egypt, May 16-18, 2004. Paper submitted.

Wooldridge, M.J. & Jennings, N.R. 1998. Pitfalls of Agent-Oriented Development. *Proceedings of the Second Conference on Autonomous Agents (Agents 98)*, ACM Press, New York. URL: http://www.ecs.soton.ac.uk/~nrj/download-files/internet-comp.pdf

Wooldridge, M., Jennings N. & Kinny D. 2000. The Gaia Methodology for Agent-Oriented Analysis and Design. *Autonomous Agents and Multi-Agent Systems*, 3: 285-312. Kluwer Academic Publishers. The Netherlands.