Towards Requirement Traceability in TROPOS

A. Castor, R. Pinto, C. Silva and J. Castro

Centro de Informática, Universidade Federal de Pernambuco, Av. Prof. Luiz Freire S/N,
Recife PE, Brazil 50732-970, +1 5581
{aop, rccp, ctlls, jbc}@cin.ufpe.br

Abstract. If we are to be successful in the development of the next generation of agent oriented systems we must deal with the critical issue of requirements traceability. Failure to do so will imply in higher costs and longer corrective and adaptable maintenance. Unfortunately most agent-oriented methodologies are not addressing this issue. Requirement traceability is intended to ensure continued alignment between stakeholders' requirements and various outputs of the system development process. In this paper we show how traceability could be applied to agent oriented development paradigm. In fact, software developers have used agents as a way to understand, model, and develop more naturally an important class of complex system. The growth of interest in software agents has recently led to the development of new methodologies based on agent concepts. However, few agent-oriented methodologies are requirement driven, or recognize traceability as an important issue to be supported. In this paper we argue that requirement traceability must be considered in agent-oriented methodologies. In particular we show how a general-purpose traceability approach can be used in the context of the Tropos framework. An e-commerce case study is used to demonstrate the applicability of the approach.

Key words: requirements traceability, agent-oriented development.

1 The Introduction

It is well known that software traceability is a significant factor of efficient software project management and software systems quality. The aim of this paper is to present some, innovative and consolidated research that supports traceability through requirements specifications, static and dynamic software design, models, system architecture models and implementation artefacts. We apply our traceability approach [1] to *Tropos*[1] approach which is requirements-driven in the sense that it is based on concepts used during early requirements analysis. To this end, Tropos adopt the concepts offered by i* [2], a modeling framework proposing concepts such as *actor* (actors can be *agents*, *positions* or *roles*), as well as social dependencies among actors including *goal*, *softgoal*, *task* and *resource* dependencies. These concepts are used for an e-commerce example[2] to model not just early requirements, but also late requirements, as well as architectural and detailed design [3, 4].

The requirement engineering process supports the understanding of the stakeholders' goals, as well as the refinement of these goals into requirements. An important task of this process is keeping track of bi-

[1] For further detail and information about *Tropos* project, see http://www.troposproject.org
[2] Based on a realistic e-commerce system development exercise of moderate complexity.

directional relationships between requirements and the development process artefacts in order to facilitate the maintenance and verification of the system [5, 6].

During *design*, traceability allows designers and maintainers to keep track of what happens when a change request is implemented before a system is redesigned. *Systems evolution* requires a better understanding of the requirements, which can only be achieved by the agility to trace back to their sources. Traceability provides the ability to cross-reference items in the requirement specifications with items in the design specifications. Moreover, t*est procedures*, if traceable to requirements or design, can be modified when errors are discovered.

It is also worth noting the important relation between traceability and configuration management. Without the latter it is impossible to trace the requirements in an appropriate manner. If the system's outputs were not well controlled it would be difficult to manage the links between them.

As a consequence of these different uses and perspectives on traceability, there are wide variations on the format and content of traceability information across different system development efforts. In fact, a reference model is needed to facilitate the construction of a requirement traceability scheme [7].

In this paper, we deal with the complexity that arises during agent-oriented development. In particular we present a general framework, which can also be useful in the context of agent-oriented development. We sketch the approach to enhance the Tropos framework to support traceability.

The structure of this paper is as follows: Section 2 presents the models that support requirement traceability and Section 3 describes the Tropos approach for agent-oriented development. In the Section 4 we apply Tropos to a case study and show all requirements traceability phases. Section 5 describes related work and finally Section 5 concludes the paper.

2. Support for requirement traceability

A general framework to support requirement traceability is presented in [7]. It includes a meta-model defining the language in which traceability models can be defined and a reference model that can be customized within the scope defined by the meta model.

In this paper requirement traceability is defined as the ability to describe and follow the life of a requirement, in both forward and backward direction, within the context of four composite, interrelated and parallel information layers: external, organisational, management and development [7]:

- *External Layer* represents, for example, constraints on the universe where the organisation is inserted.

- *Organisational Layer* represents an element (with goals and decisions) of the universe.
- *Management Layer* is related to activities such as management of people, budget and contracts that can be performed by an organization.
- *Development Layer is* related to artifacts produced by some development process.

Elements are related to one another through links with associated semantics. The notation used to represent the proposed links is based on UML (Unified Modeling Language) stereotypes.

The reference model is divided in three parts (sub-models) for clarity: Requirement Management, Design and Rationale.

- *Requirement Management sub-model*. Traceability, when implemented correctly, would greatly benefit requirement management, facilitating requirement understanding, capturing, tracking, validation and verification (Figure 1).

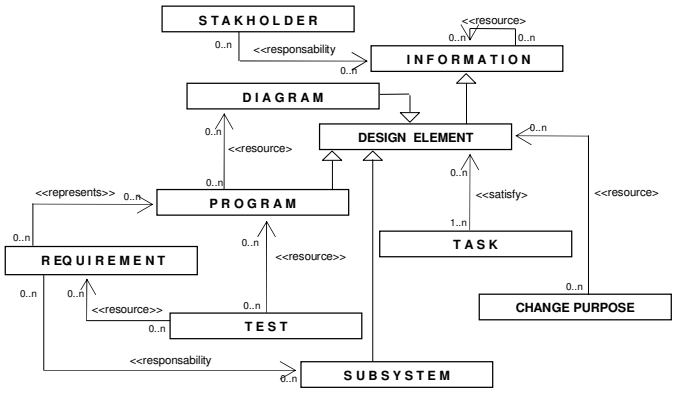

Figure 1: Requirements Management Sub-model

- *Design sub-model* is used to refer to any activity that creates artifacts, including implementation (Figure 2).

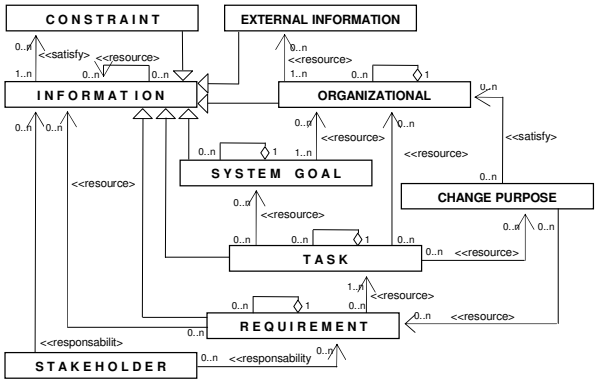

Figure 2: Design Sub-model

Beyond these sub-models, Toranzo [7] proposes a *rational model* for identification and structure of the problems and decisions made (reasoning) during the software development (Figure 3).

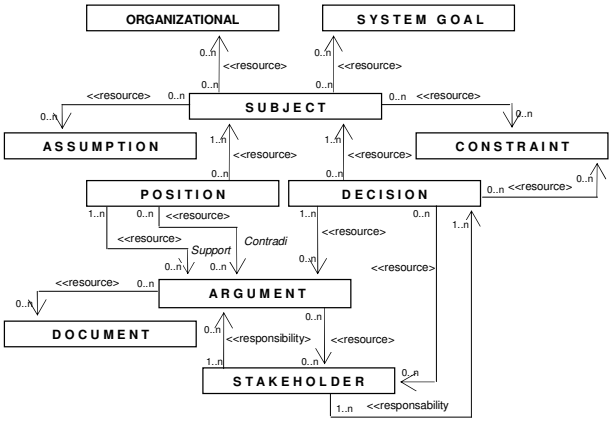

Figure 3: The Rational model

In this paper we outline a process that can be used in order to construct the models previously described. It includes three activities: Information Gathering, Information Structuring and Construction of the Traceability Matrices. The process outlined will be used in conjunction the Tropos approach (see Section 3). As such it should be applied to the following activities: Early Requirements, Late Requirements, Architectural Design and Detailed Design.

3. TROPOS

Tropos rests on the idea of using requirements modeling concepts to build a model of the system-to-be within its operational environment [3,4]. This model is incrementally refined and extended, providing a common interface to the various software development activities. The model also serves as a basis for documentation and evolution of the software system.

The proposed methodology spans four phases that can be used either following the waterfall or the spiral model respectively for sequential and iterative development [8]:

- *Early requirements*, concerned with the understanding of a problem by studying an organizational setting.
- *Late requirements*, where the system-to-be is described within its operational environment, along with relevant functions and qualities.
- *Architectural design*, where the system's global architecture is defined in terms of subsystems, interconnected through data, control and other dependencies.
- *Detailed design*, where the behavior of each architectural component is further refined.

Due to space limitation, in the sequel we only comment part of architectural design phase. An interested reader can find a full description of all phases in [3,4]. We then show how the information and decisions taken can be traced.

System architectural design has been the focus of considerable research during the last fifteen years that has produced well-established architectural styles and frameworks for evaluating their effectiveness with respect to particular software qualities.

Tropos has defined organizational architectural styles [9] for cooperative, dynamic and distributed applications such as multi-agent systems to guide the design of the system architecture.

These styles are based on concepts and design alternatives coming from research on organizational theory. From this perspective, a software system is akin to a social organization of coordinated autonomous components that interact in order to achieve specific and possibly common goals [3]. This perspective is intended to reduce as much as possible the impedance mismatch between the system and its organizational environment.

The evaluation of the styles can be done with respect to software quality attributes identified as relevant for distributed and open architectures such as multi-agent ones.

The style is based on means-ends analysis using the non-functional requirements (NFRs) framework [10]. We refine the identified requirements

to sub-requirements that are more precise and evaluate alternative organizational styles against them.

In the sequel we outline Tropos' phases through an e-business example and make some remarks of how traceability issues can be addressed.

4. Case Study

Media Shop is a store selling and shipping different kinds of media items such as books, newspapers, magazines, audio CDs, videotapes, and the like. *Media Shop* customers (on-site or remote) can use a periodically updated catalogue describing available media items to specify their order. To increase market share, *Media Shop* has decided to open up a B2C retail sales front on the Internet. With the new setup, a customer can order *Media Shop* items in person, by phone, or through the Internet. The system has been *Medi@* and is available on the world-wide-web using communication facilities provided by *Telecom Cpy*. It also uses financial services supplied by *Bank Cpy*. The basic objective for the new system is to allow an on-line customer to examine the items in the *Medi@* Internet catalogue, and place orders.

An on-line search engine allows customers with particular items in mind to search title, author/artist and description fields through keywords or full-text search. If the item is not available in the catalogue, the customer has the option of asking *Media Shop* to order it. Details about media items include title, media category (e.g., book) and genre (e.g., science-fiction), author/artist, short description, editor/publisher international references and information, date, cost, and sometimes pictures (when available).

On the next sections we describe how the traceability process previously outlined can be used in conjunction with the Tropos phases.

4.1. Early Requirements

Description provided in the previous section is sufficient for producing a first model of an organizational environment (see Figure 4). For more details, see [3].

In this phase we depict the organizational setting. *Quality Packages* is a softgoal dependence that will be stored in the EXTERNAL INFORMATION of the Requirements Management Sub-model, since it refers to an information external to the system organization. *Increase Market Share*, *Happy Customers*, *Continuing Business* goals and *Continuous Supply* softgoals are ORGANIZATIONAL INFORMATION, since these softgoals pertain to the system organisational world. *Buy Media Items*, *Consult Catalogue* and *Media Items* are REQUIREMENTS of the management layer.

The actors in the *Actor diagram for a Media Shop* (Figure 4) should be stored as STAKEHOLDER data to be linked to INFORMATION. This link is extremely important because stores information about the stakeholders and

their contributions to the system to be. When a change is required, the stakeholders in question can be questioned about possible doubts as well as conflicts can be resolved.

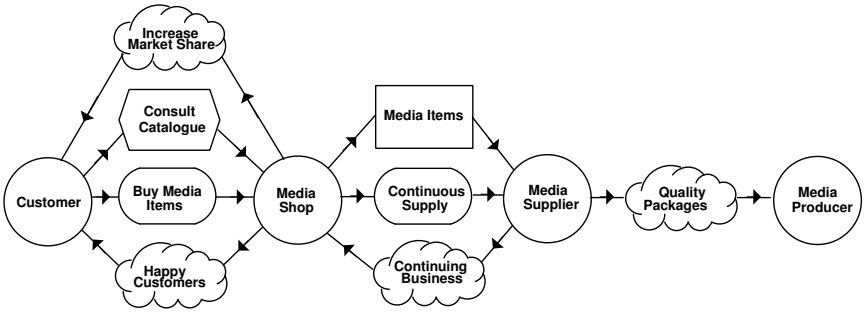

Figure 4. Actor diagram for a Media Shop

Having understood the organizational setting one can now decide to develop software system to support it.

4.2. Late Requirements Analysis

We introduce softgoal contributions to model sufficient/partial positive (respectively ++ and +) or negative (respectively -- and -) support to softgoals *Security*, *Availability*, *Adaptability*, *Attract New Customers* and *Increase Market Share*. The result of this means-ends analysis is a set of (system and human) actors who are dependees for some of the dependencies that have been postulated. For more details see [3]

In our revised example, we have included softgoals (*Availability*, *Security*, *Adaptability*) in the late requirements model. The *Availability* goal represents the ability of system agents to automatically decide at run-time which catalogue browser, shopping cart and order processor architecture fit best customer needs or navigator/platform specifications. Moreover, we could include different search engines, reflecting alternative search techniques, and let the system dynamically choose the most appropriate. The second key softgoal in the late requirements specification is *Security*. To fulfil it, we propose to support in the system's architecture a number of security strategies and let the system decide at run-time which one is the most appropriate, taking into account environment configurations, web browser specifications and network protocols used. We also require *Adaptability*, meaning that catalogue content, database schema, and architectural model can be dynamically extended or modified to integrate new and future web-related technologies.

Attract New Customer goal is one of the objective of the system so it is represented as a SYSTEM GOALs. *Availability*, *Security* and *Adaptability* softgoals or NFRs (Non-Functional Requirements) critical for the next phase (architectural design).

All tasks pictured in the Figure 5 which we have not been mentioned yet are functional requirements. All the functional and non-functional requirements are stored as REQUIREMENTS information.

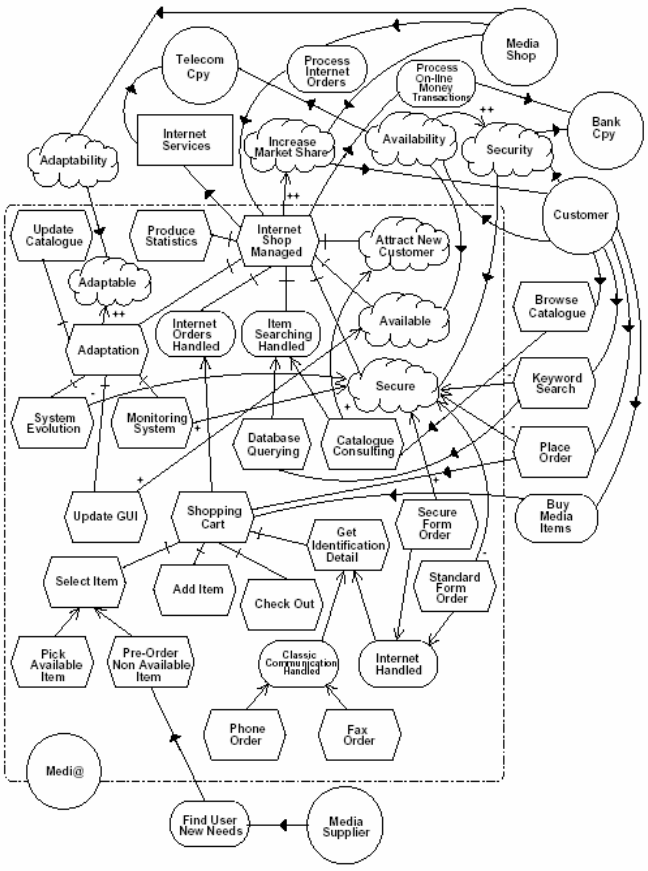

Figure 5: Rationale diagram for Medi@

Telecom Cpy and *Bank Cpy* are new stakeholders, so they are added as STAKEHOLDERS information. We have to store the *Internet Services* and *Process On-line Money Transactions* in EXTERNAL INFORMATION because both pertain to the outside world of the system but have a great impact on it.

All information identified in this phase is part of Requirements Management sub model.

Using the relationship <resource> between ORGANIZATIONAL INFORMATION and REQUIREMENTS classes (see Figure 1) we can elaborate a traceability matrix [11]. Using the matrix, we can conclude that the quantity of relationships between one requirement and all organizational information determine the main systems' requirements [11]. We can also conclude that the organizational information not related with requirements are not necessary.

In the next section we will present the traceability process applied on the Tropos architectural phase.

4.3 Architectural Design

The software quality attributes (*Availability, Security, and Adaptability*), which we highlighted in the late requirements phase, will guide the selection process of the appropriate architectural style. The Rational model captures this information. It will be useful to justify the decision taken.

To cope with non-functional requirements (software quality attributes) and select the style for the organizational setting, we go through a means-ends analysis (see [3] for more details) using the non-functional requirements (NFRs) framework [10]. We refine the identified requirements to sub-requirements that are more precise and evaluate alternative organizational styles against them (Figure 6). Considering the Rational model elements we can store on the SUBJECT element the selection process related to what organizational style will be used. The architectural styles should be represented as the POSITION for the SUBJECT. Thus for each SUBJECT there is a POSITION related to it. The notation used in NFR diagrams (++, +, --, -) to demonstrate the suitability or not of certain architecture style should be recorded as the links between the POSITIONs and each one of the ARGUMENTs. The non-functional requirements will be the ARGUMENTS for each position, because they are motivations for the decisions taken (i.e. the choice of Joint Venture architectural style). The Correlation Catalogue [9] will be stored in the CONSTRAINT element since the decision about what style will be used is limited to the using of this catalogue. The fact of choosing an architectural style based on organisational approach and not based on traditional architectural styles shall be stored in the ASSUMPTION element.

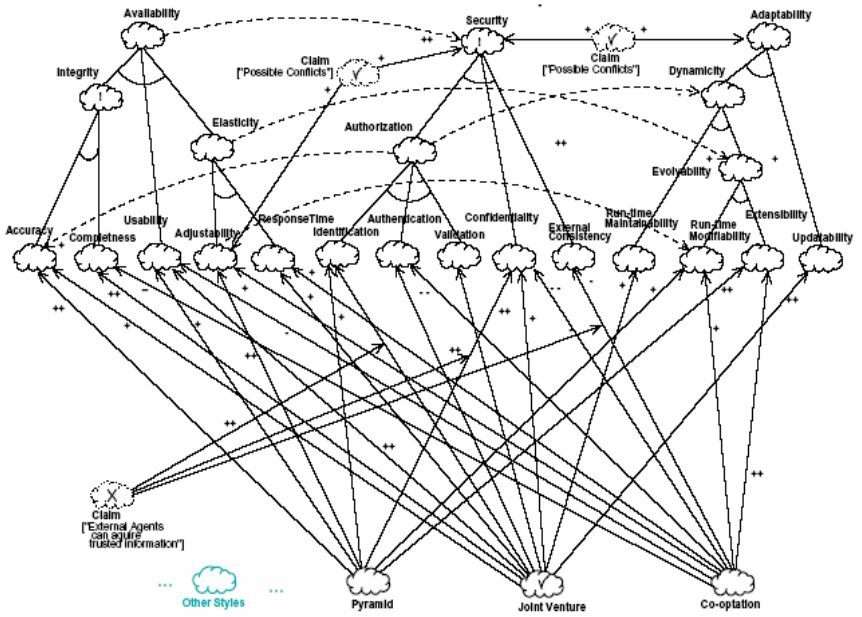

Figure 6. NFR Graph

The NFR framework links can be mapped as follows:

++ (*make)*: <support, H>
+ (*help)*: <support, M>
- (*hurt)*: <contradict, M>
-- (*break)*: <contradict, H>

Table 1 presents an example of the relationship among the POSITION and ARGUMENTS elements.

Table 1: Traceability matrix between positions and arguments

<rec>	[POS1] Pyramid	[POS2] Join Venture	[POS3] Composition
[NFR1] Availability	<support, M>	<support, H>	<contradict, M>
[NFR2] Security	<support, M>	<support, M>	<contradiz, H>
[NFR3] Adaptability	<support, M>	<support, H>	<support, M>
[NFR4] System Evolution		<support, H>	<support, M>
[NFR5] Integrity	<support, M>	<support, H>	<contradict, M>

For example it shows that the Joint Venture Style supports, in a high degree, nthe following NFR: Availability, Adaptability, System Evolution and Integrity. Whereas the NFR Security is only addressed in a moderate fashion. Similar analysis can also be made with respect to the other styles.

5. Related Work

Some agent-oriented methodologies are extensions of object-oriented methodologies (for example, Gaia [12] and MaSE [13]), while others are extensions of knowledge engineering methodologies (for example, KGR [14]).

Gaia makes an important distinction between the analysis (dealing with *abstract* concepts) and the design (dealing with *concrete* concepts) process, and provides several models to be used at each phase. In essence it constructs a society of agents, defining the role and capabilities of each individual agent, and the way the society of agents is structured.

MaSE takes an initial system specification, and produces a set of formal design documents in a graphically based style. The primary focus is to guide a designer through the software lifecycle from a prose specification to an implemented agent system. KGR consists of two viewpoints. The external viewpoint describes the social system structure and dynamics. It includes an Agent Model and an Interaction Model. The internal viewpoint is composed of three models: the Belief Model, the Goal Model, and the Plan Model. These models specify how an agent perceives the environment and how it chooses its actions based on this perception.

The comparison of these methodologies is out of scope [15] but we agree that none of them support requirement traceability explicitly. However some of them are more easily adaptable because they store some links between their elements.

6. Conclusions

Requirement traceability has been recognized by many as an important pre-requisite for developing and maintaining high quality software. In this work we argue that the Tropos framework can be extended to address requirements traceability. Our traceability approach is able to record information from all phases supported by Tropos approach.

The benefits of requirements traceability are manifold: software quality can be improved since we can check if all stakeholders requirements are addressed by the system. Similarly, an impact analysis can also be performed before the implementation of a change request. This is possible because we requirements impacted by the change can be detected as well as the links between these requirements and other system components, like design and

implementation,. Hence the change and make effort estimates become more accurate and consequently we can minimize the time and cost of software maintenance.

Further work is still required to make a comparison between issues of requirement traceability in engineering of object–oriented systems versus multi agent systems, i.e. in what way the requirement traceability methods differ and the reasons. Proper tool support is also another topic that needs to be addressed.

7. References

1. J. Castro, R. Candida, A. M. Castor, and J. Mylopoulos, "Requirements Traceability in Agent Oriented Software Engineering". Book chapter In Software Engineering for Large-Scale Multi-Agent Systems: Research Issues and Practical Applications, LNCS 2603, Editors A. Garcia; C. Lucena; F. Zambonelli; A. Omicini; J. Castro Springer Verlag. 2003.
2. E. Yu, *Modelling Strategic Relationships for Process Reengineering*. PhD thesis, University of Toronto, Department of Computer Science, 1995.
3. J. Castro, M. Kolp, and J. Mylopoulos, *Towards Requirements-Driven Information Systems Engineering: The Tropos Project*. Information Systems Journal, Elsevier, 2002. Vol 27, pp. 365-89
4. P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos, A. Perini "Tropos: An Agent –Oriented Software Development Methodology'", in *Autonomous Agents and Multi –Agent Systems* 8 (3): 203-236, May 2004.
5. B. Ramesh, and M. Jarke, *Towards Reference Models For Requirements Traceability*. IEEE Transactions on Software Eng., vol. 27, pp. 58-93, Jan. 2001
6. O. Gotel, *Contribution Structures for Requirements Engineering*. Ph.D Thesis. Department of Computing, Imperial College of Science, Technology, and Medicine, London, U.K., 1996.
7. M. Toranzo, *A Framework to Improve Requirements Traceability* (in Portuguese: Um Framework para Melhorar o Rastreamento de Requisitos). Ph.D thesis, Centro de Informática daUniversidade Federal de Pernambuco – UFPE, Brazil, December, 2002.
8. P. Kruchten. The Rational Unified Process: An introduction. Addison-Wesley, 2003.
9. T. T. Do, M. Kolp and A. Pirotte. "Social Patterns for Designing Multi-Agent Systems", in *Proceedings of the 15th International Conference on Software Engineering and Knowledge Engineering (SEKE 2003)*, San Francisco, USA, July 2003.
10. L. K. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos, *Non-Functional Requirements in Software Engineering*, Kluwer Publishing, 2000.
11. A. Castor, "Requirements Traceability on the Agent Oriented Development Process". (in Portuguese: Rastreamento de Requisitos no Processo de Desenvolvimento Orientado a Agentes). Ph.D dissertation, Centro de Informática da Universidade Federal de Pernambuco – UFPE, Brazil, August 2004.
12. M. Wooldridge, N. Jennings, and D. Kinny *The Gaia Methodology for Agent-Oriented Analyis and Design*, Journal of Autonomous Agents and Multi-Agent Systems, 2000.
13. M. Wood, and S. A. DeLoach, "An Overview of the Multiagent System Engineering Methodology", in the *First International Workshop on Agent-Orientes Software Engineering (AOSE-2000)*, June, 10, 2000 – Limerick. Ireland
14. D. Kinny, M. Georgeff, and A. Rao, "A Methodology and Modelling Technique for Systems of BDI Agents", in W. Van Der Velde and J. Perram, editors. Agents Breaking Away: Proceedings of the Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World MAAMAW'96, (LNAI Volume 1038). Springer-Verlag, 1996.
15. C. T. L. L. Silva, P. C. Tedesco, J. B. F. Castro, R. C. C. Pinto, "*Comparing Agent-Oriented Methodologies Using NFR Approach*", in Proceedings of the SELMAS 2004 - Software Engineering for Large-Scale Multi-Agent Systems, Edinburgh, Scotland, 2004.