The Specification of Requirements in the MADAE-Pro Software Process

Rosario Girardi, Adriana Leite

Department of Computer Science
Federal University of Maranhão (UFMA) – São Luís, MA – Brazil
`rosariogirardi@gmail.com, adri07lc@gmail.com`

Abstract. *MADAE-Pro is an ontology-driven process for multi-agent domain and application engineering which promotes the construction and reuse of agent-oriented applications families. This article introduces MADAE-Pro, emphasizing the description of its domain analysis and application requirements engineering phases and showing how software artifacts produced from the first are reused in the last one. Illustrating examples are extracted from two case studies we have conducted to evaluate MADAE-Pro. The first case study assesses the Multi-Agent Domain Engineering sub-process of MADAE-Pro through the development of a multi-agent system family of recommender systems supporting alternative (collaborative, content-based and hybrid) filtering techniques. The second one, evaluates the Multi-Agent Application Engineering sub-process of MADAE-Pro through the construction of InfoTrib, a Tax Law recommender system which provides recommendations based on new tax law information items using a content-based filtering technique. ONTOSERS and InfoTrib were modeled using ONTORMAS, a knowledge-based tool for supporting and automating the tasks of MADAE-Pro.*

1. Introduction

MADAE-Pro ("Multi-agent Domain and Application Engineering Process") is a knowledge-based process for the development and reuse of families of multi-agent software systems.

A family of software systems is defined as a set of systems sharing some commonalities but also having particular features [Czarnecki, K. and Eisenecker, U. W. 2000]. The agent-oriented software community has increased its interest in this kind of product considering not only its already known potential for improving the quality of software applications and for increasing the productivity of software development [Pohl,K., Bockle, G. and Linden, F. 2005]. Also, agent-oriented software families are nowadays feasible because of the maturity and experience gained on agent-oriented software development.

A software development process is a model that specifies a *life cycle*, describing the phases through which transits a software product from its conception through its development along with a *methodology* that integrates the *techniques* to be applied in each one of the phases according to a particular development paradigm.

MADAE-Pro consists of two complementary sub-processes:

- Multi-agent Domain Engineering, a process for the development of a family of multi-agent software systems in a problem domain, by applying MADEM ("Multi-agent Domain Engineering Methodology")[Girardi, R. and Marinho, L. 2007]; and
- Multi-agent Application Engineering, the process for constructing a specific agent-oriented application by reusing one or more of those families, using MAAEM ("Multi-agent Application Engineering Methodology") [Drumond, L. and Girardi, R. 2008] [Leite, A., Girardi, R. and Cavalcante, U. 2008b].

The process consolidates a long term research effort on techniques, methodologies and tools for promoting reuse on agent-oriented software development.

The software products generated in each task of MADAE-Pro are represented as instances of the ONTORMAS knowledge base. ONTORMAS ("ONTOlogy driven tool for the Reuse of Multi-Agent Systems") [Leite, A., Girardi, R. and Cavalcante, U. 2008a] is a knowledge-based tool for supporting and automating the MADAE-Pro tasks. ONTORMAS is an extension of ONTOMADEM ("A Knowledge-based Tool for Multi-Agent Domain Engineering") [Girardi, R., Leite, A. 2008], a tool which supports just the MADEM methodology.

This work introduces MADAE-Pro emphasizing the description of its domain analysis and application requirements engineering phases, illustrating how software artifacts produced from the first phase are reused in the last one. Examples are extracted from two case studies we have conducted to evaluate the process [Mariano, R. at al. 2008] [Mariano, R. 2008]. The first case study evaluates the Multi-Agent Domain Engineering sub-process of MADAE-Pro through the development of ONTOSERS (ONTOlogy-based SEmantic web Recommender Systems"), a multi-agent system family of recommender systems supporting alternative (collaborative, content-based and hybrid) filtering techniques. The second one, evaluates the Multi-Agent Application Engineering sub-process of MADAE-Pro through the reuse of ONTOSERS family for the development of InfoTrib. InfoTrib [Mariano, R. 2008] is a tax law recommender system in which, based on a user profile specifying his/her interests in the diverse types of taxes, the system provides recommendations based on new tax law information items, using a content-based filtering technique. The modeling process revealed being consistent and capable of generating products with high potential of reuse. The ONTOSERS family provided an appropriate framework for experimentation, analysis and evaluation of diverse information filtering algorithms. INFOTRIB a tax law recommender system was developed through the reuse of the ONTOSERS family. [Mariano, R., 2008].

The paper is organized as follows. Section 2 describes the MADAE-Pro software development process. Section 2.1 introduces its lifecycle and a general description of the support that the MADEM and MAAEM methodologies provide to each one of its phases. Section 2.2 gives an overview of the ONTORMAS tool. Section 3 details the particular tasks of the Multi-agent Domain Analysis and Multi-agent Application Requirements Engineering phases of MADAE-Pro along with the guidelines provided by these methodologies to carry out those tasks. Examples from case studies conducted for the evaluation of these phases are also described. Section 4 references related work discussing its similarities and differences with MADAE-Pro. Finally, section 5 concludes the paper with some considerations on ongoing work.

2. The MADAE-Pro Software Process Model

MADAE-Pro is a knowledge-based process model which integrates an iterative, incremental and goal-driven life cycle (see section 2.1) along with the MADEM and MAAEM methodologies for Multi-agent Domain Engineering and Multi-agent Application Engineering, respectively. Its phases, tasks and products are conceptualized in the ONTORMAS knowledge-base and both, specific or multi-agent system families are represented as instances of this knowledge base (see section 2.2). Main modeling concepts and tasks of MADEM and MAAEM are based both on techniques for Domain and Application Engineering [Arango, G. 1988], [Czarnecki, K. and Eisenecker, U. W. 2000] [Girardi, R. 1992] [Harsu, M. 2002][Pohl,K., Bockle, G. and Linden, F. 2005] and for development of multi-agent systems [Bresciani, P. et al. 2004], [Cossentino, M. et al. 2004] [Dileo, J., Jacobs, T. and Deloach, S. 2002] [Perini, A. and Susi, A. 2004] [Odell, J., Parunak, H. V. D. and Bauer, B. 2000].

The semantic network shown in Figure 1 represents the main elements involved in MADAE-Pro: the MADEM and MAAEM methodologies; the techniques GRAMO (Generic Requiremente Analysis Method based on Ontologies), DDEMAS (Domain Design technique for Multi-Agent Systems) and DIMAS (Domain Design technique for Multi-Agent Systems) which integrate the MADEM methodology, and are associated, respectively, to the phases of Domain Analysis, Domain Design and Domain Implementation; the techniques SRAMO (Specific Requirement Analysis Method based on Ontologies), ADEMAS (Aplication Design technique for Multi-Agent Systems) and AIMAS (Application Implementation Technique for Multi-Agent Systems) which are part of the MAAEM methodology and are associated, respectively, to the phases of Application Requirements Engineering, Application Design and Application Implementation; the adopted life cycle, which is iterative and incremental; the ONTORMAS tool, which is used to guide the development tasks, perform visual modeling, document and store the artifacts produced during the process execution; and finally, the modeling language for multi-agent systems, MADAE-ML. This language provides a graphical representation for models and modeling concepts of the MADAEM and MAAEM methodologies and for roles in the process (e.g. Programmer, System Analyst), responsible for the realization of one or more tasks during the execution process.

For the specification of a problem to be solved, both methodologies focus on modeling goals, roles and interactions of entities of an organization, representing the requirements of either a multi-agent system family or a specific multi-agent application from the point of view of the organization stakeholders.

An organization is composed of both passive and active entities. Active entities have knowledge and use it to exhibit autonomous behavior performed in order to achieve specific goals. The achievement of specific goals allows reaching the general goal of the organization. For instance, an information system can have the general goal of “satisfying the information needs of an organization” and the specific goals of “satisfying dynamic or long term information needs”.

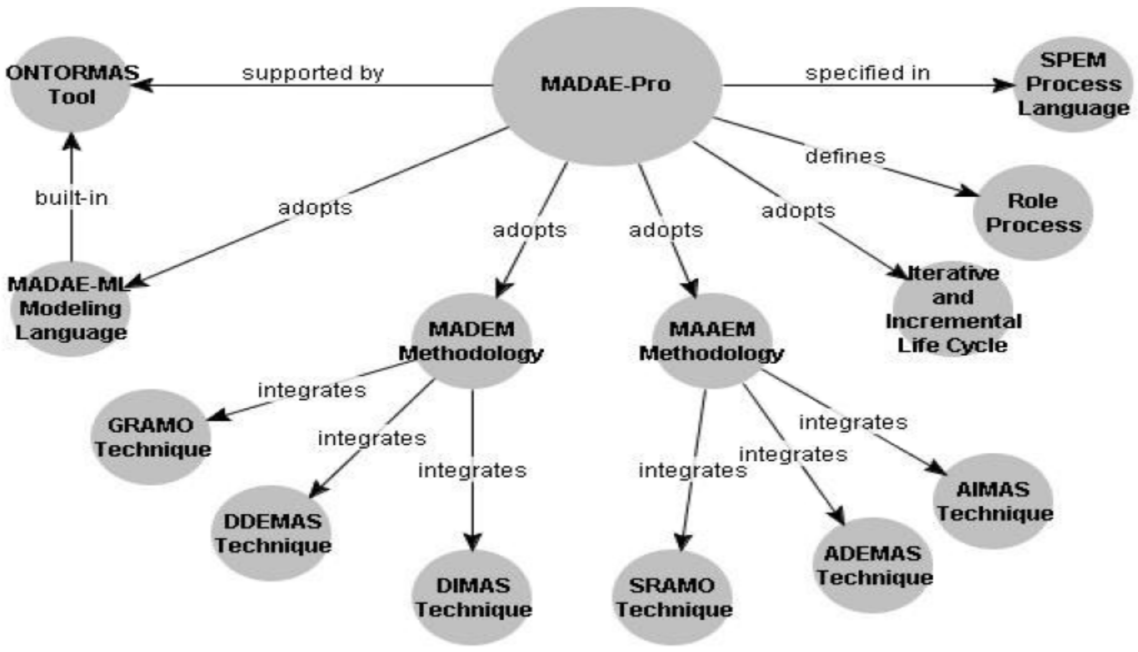


Figure 1 - Main concepts involved in MADAE-Pro

Specific goals are reached through the performance of responsibilities in charge of particular roles with a certain degree of autonomy. Pre-conditions and post-conditions may need to be satisfied for/after the execution of a responsibility. Knowledge can be consumed and produced through the execution of a responsibility. For instance, an entity can play the role of “retriever” with the responsibility of executing the responsibility of satisfying the dynamic information needs of an organization. Another entity can play the role of “filter”, in charge of the responsibility of satisfying the long-term information needs of the organization. Sometimes, entities have to communicate with other internal or external entities (like stakeholders) to cooperate in the execution of a responsibility. For instance, the entity playing the role of “filter” may need to interact with a stakeholder to observe his/her behavior in order to infer his/her profile of information interests.

For the specification of a design solution, roles are assigned to reactive or deliberative agents structured and organized into a particular multi-agent architectural solution according to non-functional requirements.

Agents have skills related to one or a set of computational techniques that support the execution of responsibilities in an effective way. According to the previous examples, skills can be, for instance, the rules of the organization to access and structure its information sources.

For the implementation, the agent design models are mapped to agents, behaviors and communication acts, concepts involved in the JADE framework [Bellifemine, F. et al. 2003] and JESS [Friedman-Hill, E. 2003], which are the adopted implementation platform. This platform was chosen for being one of the few public domain platforms available allowing the construction of deliberative agents; because of its popularity and maturity; and ease of integration with the Protégé platform [Gennari, J. et al. 2002], frequently used to build ontologies by the research group. JADE is a middleware for the development and run-time execution of peer-to-peer applications which are based on the agents paradigm, and JESS is a rule engine and scripting environment that allows build software that has the capacity to "reason" using knowledge supplied in the form of declarative rules. Goals, roles, and responsibilities are the modeling abstractions of the system requirements which are mapped to agents, behaviors and communication acts to construct an agent-oriented computational solution satisfying such requirements.

Variability modeling is a main concern on the construction of multi-agent system families. In MADAE-Pro, it is carried out in parallel with all MADEM phases to determine the common and variable parts of a family. This is done by identifying the "Variation Points" and its correspondent "Variants". A variation point is the representation of a concept subjected to variation. A variant represents the alternative or optional variations of such a concept.

2.1. The MADAE-Pro lifecycle

Figure 2 illustrates the MADAE-Pro process life cycle using the SPEM ("Software Process Engineering Metamodel") notation [SPEM 2010]. The cycle is iterative, incremental and goal-driven. Development is carried out through successive increments, looking for reducing software complexity. It is initiated with the decision of development of a new family of applications, or a specific one, by specifying a new general goal and restarted for the development of a new specific goal or to update an existing one in evolutive and corrective maintenance, respectively ("new or existing goal' in diamond of Figure 2).

Iterations can also occur between the phases for refining modeling products. Techniques are associated to each development phase to guide the modeling tasks. In Domain Engineering, the techniques GRAMO, DDEMAS and DIMAS guide, respectively, the tasks of the Domain Analysis, Domain Design and Domain Implementation phases. In Application Engineering, the techniques SRAMO, ADEMAS and AIMAS guide, respectively, the tasks of the Application Requirements Engineering, Application Design and Application Implementation phases. Figure 2 also shows the consumed and generated products of each phase. MADAE-Pro consists of six development phases: domain analysis, domain design and domain implementation, supported by the MADEM methodology; and application requirements engineering, application design and application implementation, guided by the MAAEM methodology.

2.1.1 The MADEM phases

The domain analysis phase of MADEM approaches the construction of a domain model specifying the current and future requirements of a family of applications in a domain by considering domain knowledge and development experiences extracted from domain specialists and applications already developed in the domain, including products of the Multi-agent Application Engineering sub-process.

This phase consists of the following modeling tasks: modeling of domain concepts, goal modeling, role modeling, role interaction modeling and user interface prototyping. The product of this phase, a domain model, is obtained through the composition of the products constructed through these tasks: a concept model, a goal model, a role model, a set of role interaction models, one for each specific goal in the goal model and a prototype of the user interface. Next section details the domain analysis tasks and products.

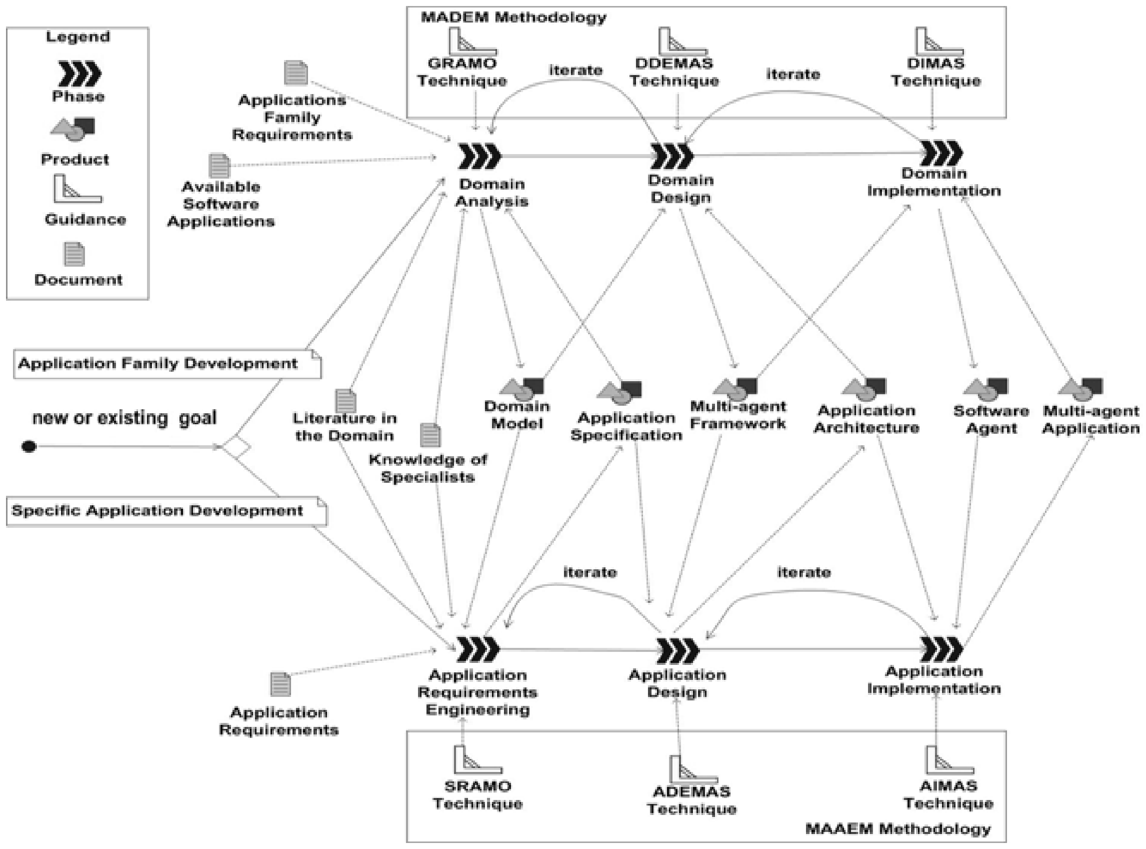


Figure 2. The MADAE-Pro Lifecycle

The domain design phase of MADEM approaches the architectural and detailed design of multi-agent frameworks providing a solution to the requirements of a family of multi-agent software systems specified in a domain model. This phase consists of two sub-phases: the architectural design sub-phase which establishes an architectural model of the multi-agent society including the knowledge shared by all agents in their communication and their coordination and cooperation mechanisms; and the agent design sub-phase which defines the internal design of each reactive or deliberative agent, by modeling its structure and behavior. A Multi-agent Framework Model of the Multi-agent Society is constructed as a product of this phase, composed of a Multi-agent Society Knowledge Model, an Architectural Model and a set of Agent Models.

The domain implementation phase of MADEM approaches the mapping of design models to agents, behaviors and communication acts, concepts involved in the JADE/JESS framework [Bellifemine, F. et al. 2003][Friedman-Hill, E. 2003], which is the adopted implementation platform. An implementation model of the multi-agent society is constructed as a product of this phase, composed of a model of agents and behaviors and a model of communication acts.

2.1.2 The MAAEM phases

MAAEM is a methodology for requirement analysis, design and implementation of multi-agent applications through compositional reuse of software artifacts such as domain models, multi-agent frameworks, pattern systems and software agents previously developed in the MADEM Domain Engineering process.

The requirements analysis phase of MAAEM looks for identifying and specifying the requirements of a particular application by reusing requirements already specified in domain models. This phase follows a set of modeling tasks consistently uniform with the ones of the MADEM domain analysis phase, for producing a set of models composing the multi-agent requirements specification of the application. The MAAEM requirements analysis phase is performed through the following modeling tasks: concept modeling, goal modeling, role modeling, role interaction modeling and user interface prototyping. The product of this phase, an application specification, is obtained through the composition of the products constructed through these tasks: a concept model, a goal model, a role model, a set of role interaction models, one for each specific goal in the goal model and a prototype of the user interface. Next section details the requirements analysis tasks and products.

In the application design phase, developers reuse design solutions of a family of applications and adapt them to the specific requirements of the application under development. A set of models composing the multi-agent application architecture are produced by following a set of modeling tasks consistently uniform with the ones of the MADEM domain design phase. This phase consists of two tasks: the Architectural Design task aiming at constructing a multi-agent society architectural model and the Agent Design task, which defines the internal structure of each reactive or deliberative agent in the society. The Architectural Design task consists of four sub-tasks: Multi-agent Society Knowledge Modeling, Multi-Agent Society Modeling, Agent Interaction Modeling, and Coordination and Cooperation modeling.

In the application implementation phase, agent behaviors and interactions are identified and specified in a particular language/platform for agent development. A Behaviors Model and Communication Acts Model are generated in this development phase.

Along all MAAEM phases, reuse is carried out by identifying variation points in MADEM products and selecting appropriate variants.

2.2. The ONTORMAS Tool

ONTORMAS [Leite, A., Girardi, R. and Cavalcante, U. 2008a] is a knowledge-based system whose knowledge base is an ontology which conceptualizes the MADAE-Pro methodologies. It guides the modeling tasks and representation of their generated products as instances of its class hierarchy.

Ontologies [Gruber, T. R. 1995] provide an unambiguous terminology that can be shared by all involved in a software development process. They can also be as generic as needed allowing its reuse and easy extension. These features turn ontologies useful for representing the knowledge of software engineering techniques and methodologies, and an appropriate abstraction mechanism for the specification of high-level reusable software artifacts like domain models, frameworks and software patterns.

ONTORMAS was developed in a two phase development process: the specification and the design of the ontology. In the specification phase, a conceptualization of MADEM and MAAEM where represented in a semantic network. In the design phase, concepts and relationships in the semantic network were mapped to a frame-based ontology in Protégé [Gennari, J. et al. 2002]. A graphical notation was defined for the representation of each modeling product.

The ONTORMAS ontology consists of a set of classes organized hierarchically, with the main super classes (Figure 3): "Variable Concepts," "Modeling Concepts," "Modeling Tasks" and "Modeling Products". The super class "Variable Concepts" and corresponding subclasses are used to specify the variability of a multi-agent system family. This is accomplished through the definition of “Variation Points" and "Variants". A variation point represents a variable concept. A variant represents the alternative or optional variations of such concept. The super class "Modeling Concepts" specifies the modeling concepts of the MADEM and MAAEM methodologies. In the super class "Modeling Tasks" and corresponding subclasses, the MADEM and MAAEM modeling tasks are defined.

As an example, Figure 4 illustrates the representation of the tasks performed in the phases of Domain Analysis and Application Requirements Analysis. These tasks consist of the "Domain Engineering Tasks", which subtasks are related to the MADEM methodology and the "Application Engineering Tasks", related to the MAAEM methodology. The super class "Modeling Products" and corresponding subclasses define the MADEM and MAAEM products. Products can be simple or composed of sub-products. For instance, Figure 5 illustrates the classes and instance examples of the goal models produced by both MADEM and MAAEM.

The products of MADEM and MAAEM are represented as instances of the corresponding concepts in the ONTORMAS class hierarchy, having each modeling concept a particular graphical notation. This facilitates not only the instantiation process but also contributes for reducing the complexity of the modeling tasks allowing the visualization, decomposition and refinement of the modeling products. Figure 6 illustrates the creation of the ONTOSERS domain model and their respective sub-products. For that it was required the instantiation of the "Modeling Tasks" sub-classes ("Concept Modeling", "Goal Modeling," "Role Modeling", "Role Interaction Modeling" and "User Interface Prototyping") and the corresponding "Modeling Products” sub-classes ("Concept Model", "Goal Model", "Role Model", "Role Interaction Models" and "Prototype of the User Interface").

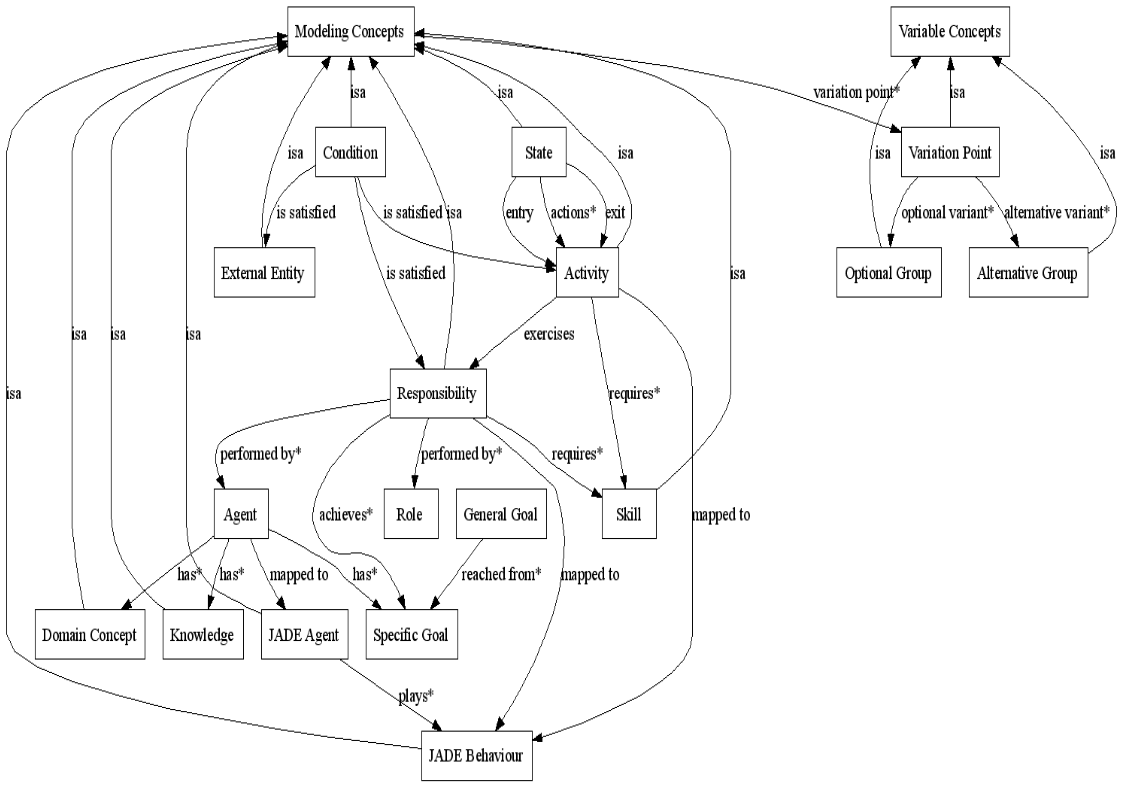


Figure 3. Semantic network illustrating main modeling concepts of MADEM and MAAEM

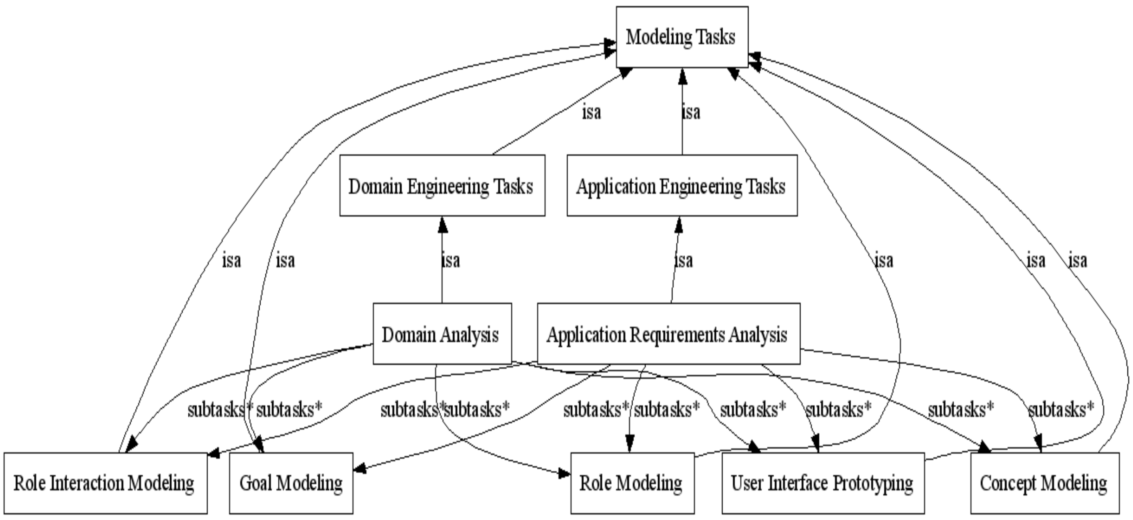

Modeling Tasks
Domain Engineering Tasks
Application Engineering Tasks
Domain Analysis
Application Requirements Analysis
Role Interaction Modeling
Goal Modeling
Role Modeling
User Interface Prototyping
Concept Modeling
isa
subtasks*

Figure 4. Semantic Network of the tasks and subtasks of the Analysis Phase of MADEM and MAAEM

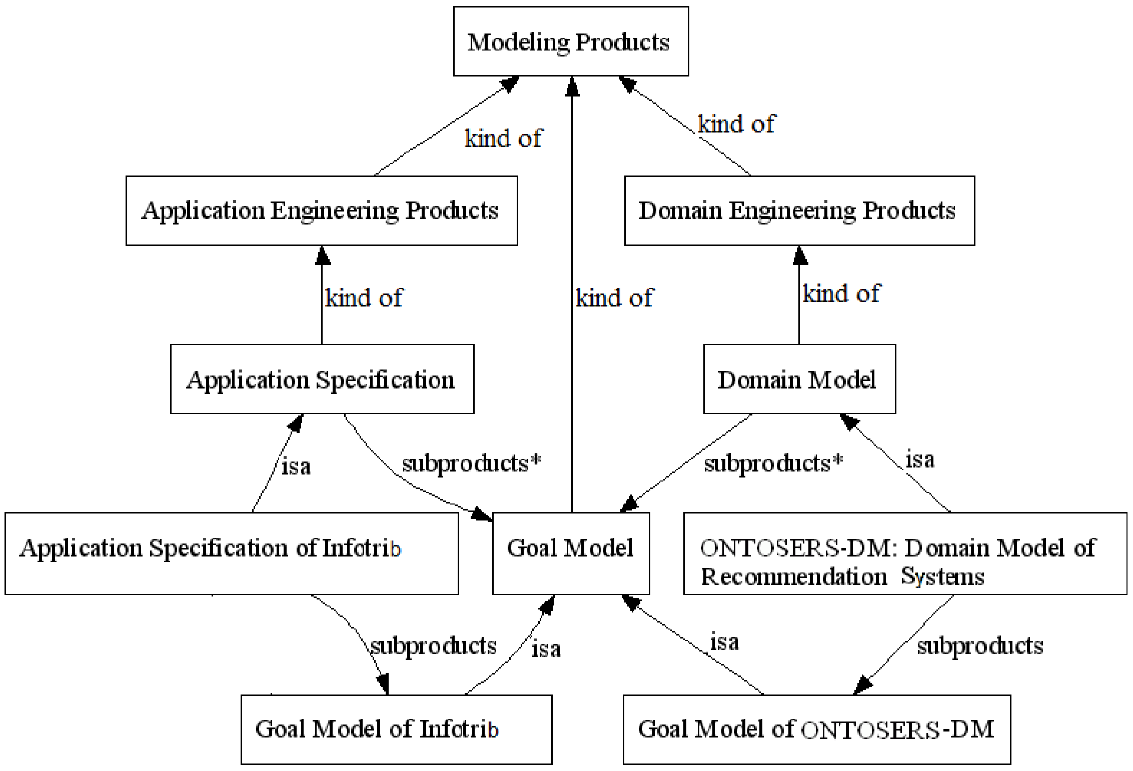
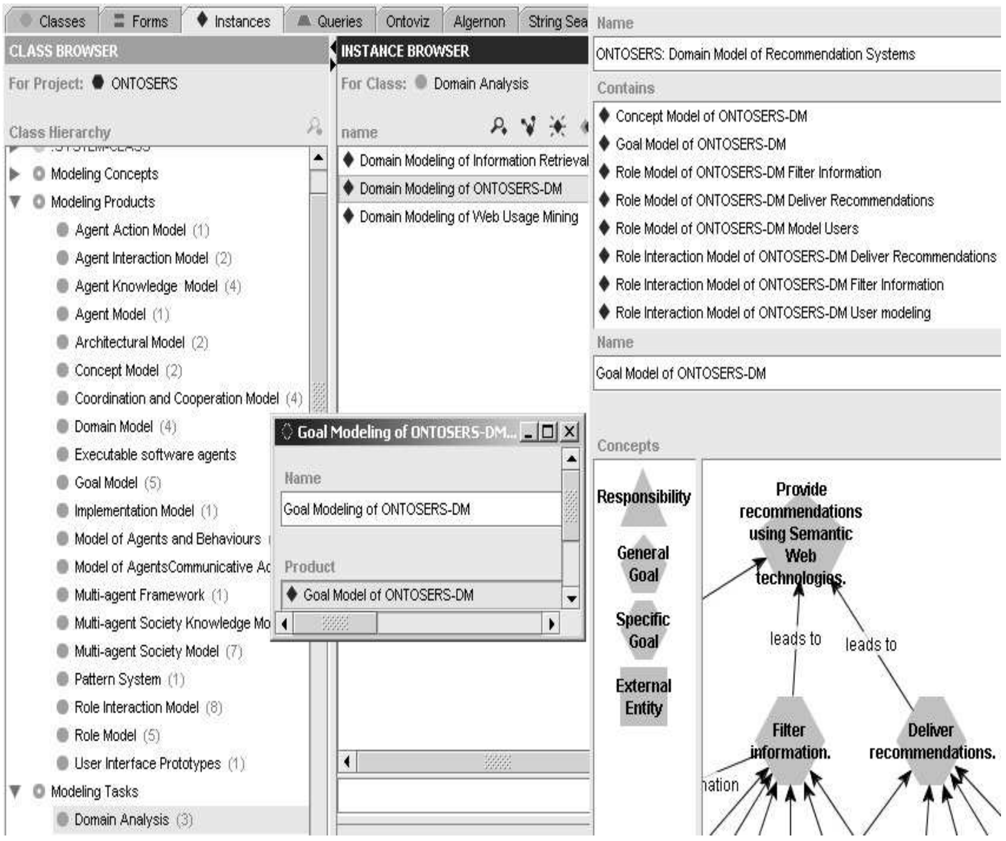

Modeling Products
kind of
kind of
Application Engineering Products
Domain Engineering Products
kind of
kind of
kind of
Application Specification
Domain Model
isa
subproducts*
subproducts*
isa
Application Specification of Infotrib
Goal Model
ONTOSERS-DM: Domain Model of Recommendation Systems
subproducts
isa
isa
subproducts
Goal Model of Infotrib
Goal Model of ONTOSERS-DM

Figure 5. Relationships between classes and instances of modeling products



For Project: ● ONTOSERS

Class Hierarchy

:SYSTEM-CLASS
- Modeling Concepts
- Modeling Products
  - Agent Action Model (1)
  - Agent Interaction Model (2)
  - Agent Knowledge Model (4)
  - Agent Model (1)
  - Architectural Model (2)
  - Concept Model (2)
  - Coordination and Cooperation Model (4)
  - Domain Model (4)
  - Executable software agents
  - Goal Model (5)
  - Implementation Model (1)
  - Model of Agents and Behaviours
  - Model of AgentsCommunicative Ac
  - Multi-agent Framework (1)
  - Multi-agent Society Knowledge Mo
  - Multi-agent Society Model (7)
  - Pattern System (1)
  - Role Interaction Model (8)
  - Role Model (5)
  - User Interface Prototypes (1)
- Modeling Tasks
  - Domain Analysis (3)

For Class: ● Domain Analysis

name
- Domain Modeling of Information Retrieval
- Domain Modeling of ONTOSERS-DM
- Domain Modeling of Web Usage Mining

Goal Modeling of ONTOSERS-DM...

Name
Goal Modeling of ONTOSERS-DM

Product
- Goal Model of ONTOSERS-DM

Name
ONTOSERS: Domain Model of Recommendation Systems

Contains
- Concept Model of ONTOSERS-DM
- Goal Model of ONTOSERS-DM
- Role Model of ONTOSERS-DM Filter Information
- Role Model of ONTOSERS-DM Deliver Recommendations
- Role Model of ONTOSERS-DM Model Users
- Role Interaction Model of ONTOSERS-DM Deliver Recommendations
- Role Interaction Model of ONTOSERS-DM Filter Information
- Role Interaction Model of ONTOSERS-DM User modeling

Name
Goal Model of ONTOSERS-DM

Concepts

Figure 6. Instantiation Process of ONTORMAS for the ONTOSERS Domain Model creation

3. The Domain Analysis and Application Requirements Engineering Tasks

This section describes the Domain Analysis and Application Requirements Engineering tasks of MADAE-Pro showing how the software artifacts of the ONTOSERS domain model [Mariano, R. at al. 2008] are produced and reused on the development of the InfoTrib multi-agent recommender system [Mariano, R. 2008].

ONTOSERS-DM is a domain model that specifies the common and variable requirements of recommender systems based on the ontology technology of the Semantic Web [Shadbolt, Hall and Berners-Lee 2006], using three informaton filtering approaches: content-based (CBF), collaborative (CF) and hybrid filtering (HF). InfoTrib is a tax law recommender system in which, based on a user profile specifying his/her interests in the diverse types of taxes, the system provides recommendations based on new tax law information items.

Figure 7 shows a refinement of the MADAE-Pro lifecycle, detailing the tasks and products of the Domain Analysis (see Section 3.1) and Application Requirements Engineering phases (see Section 3.2).

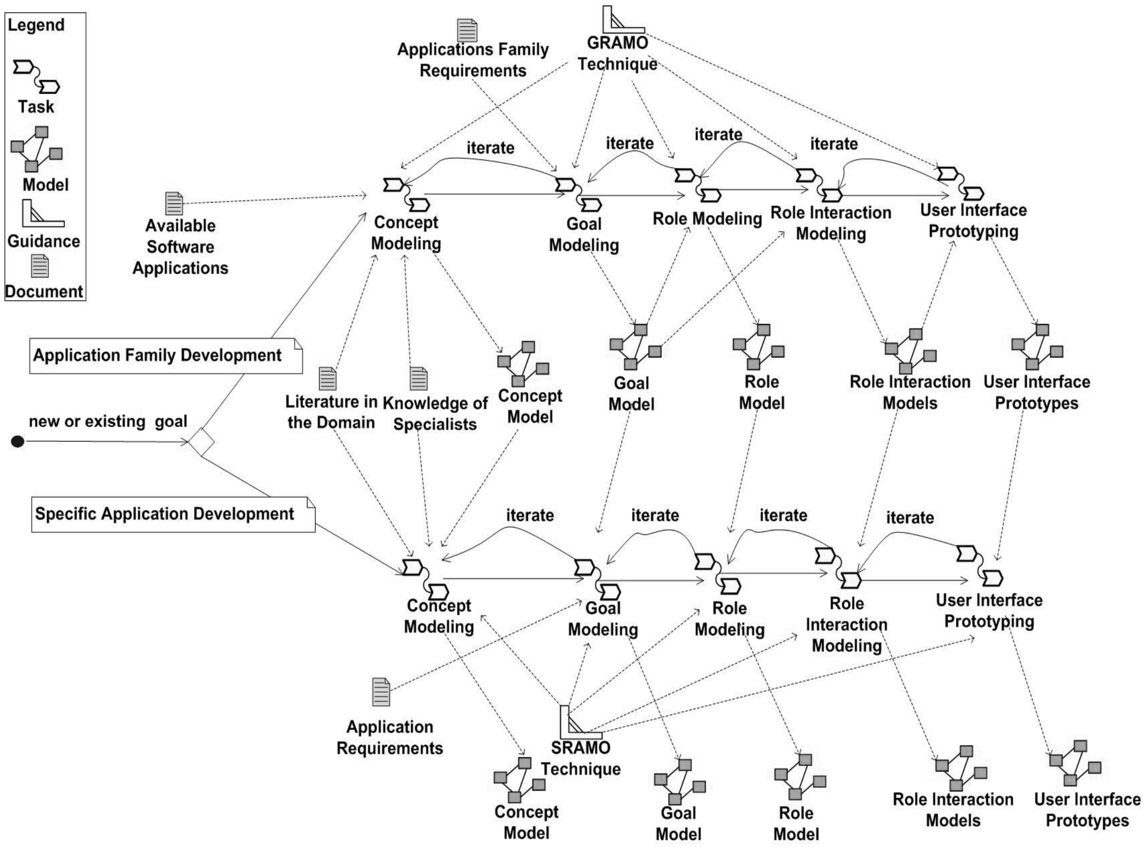


Figure 7. The Domain Analysis and Application Requirements Engineering Phases of MADAE-Pro

3.1. The Domain Analysis Tasks of MADAE-Pro

The concepts modeling task aims at just performing a brainstorming of domain concepts and their relationships, representing them in a concept model.

The purpose of the goal modeling task is to identify the common and variant goals of the family of systems, the stakeholders with which it cooperates and the responsibilities needed to achieve them. Its product is a goal model, specifying the general and the system family hierarchy of specific goals along with the stakeholders, responsibilities and variant groups. In this task, variability modeling looks for identifying variant points in specific goals related with variant groups of responsibilities.

As an example, Figure 8 represents the goal model of ONTOSERS. The "Provide Recommendations using Semantic Web Technology" general goal is reached through the "Model Users", "Filter Information" and "Deliver Recommendations" specific goals. In order to achieve the "Filter Information" specific goal, it is necessary to perform the "Ontology Instance User Model Creation and Update" responsibility, which also contributes to reach the "Model Users" specific goal. Besides that, the "Grouping of user models", "Information Items based on Ontology Instance Representation" and "Similarity Analysis" responsibilities are needed. The "Grouping of Users Models" responsibility allows for identifying groups of users with similar interests.

The "Model Users" specific goal has a variation point with groups of responsibilities for user profile acquisition, being possible to choose between three alternative variants: "Implicit Profile Acquisition", "Explicit Profile Acquisition" or both. The last responsibility, "Ontology Instance User Model Creation and Update" is fixed, i.e. it is required in all the applications of the family. The "Filter Information" specific goal has a variation point that has as variant alternatives: the "Grouping of users models" responsibility, required in systems that use CF; and the "Information Items based on Ontology Instance Representation" responsibility required in the ones using CBF. The "Deliver Recommendations" specific goal does not have variation points, therefore the "Similarity Analysis", "Personalized Recommendations Production" and "Delivery of Personalized Recommendations" responsibilities are required in all the applications of the family, then belonging to the fixed part of the goal model.

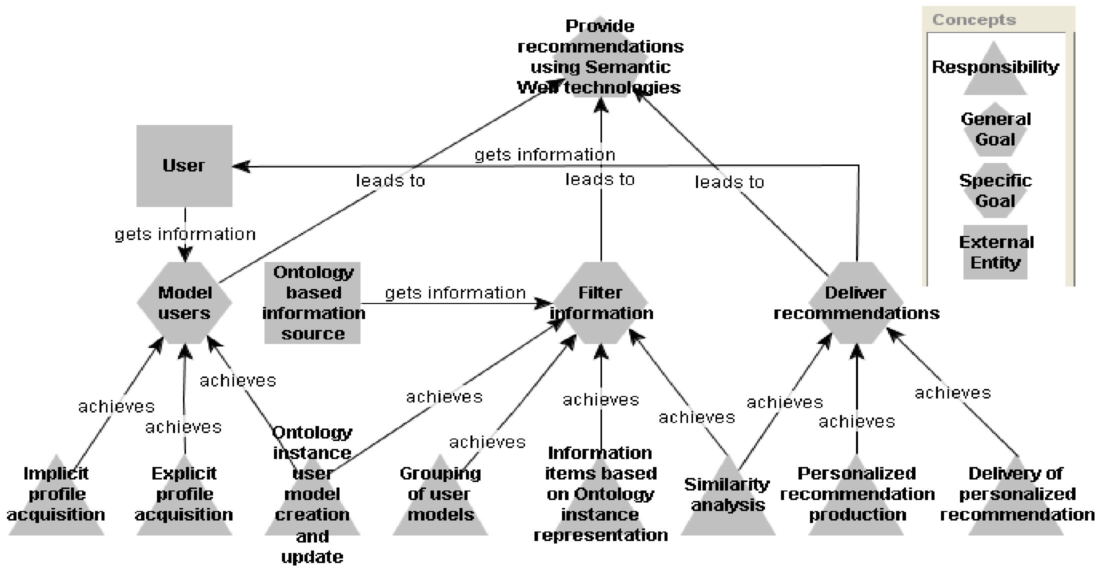

Provide recommendations using Semantic Web technologies
Concepts
Responsibility
General Goal
Specific Goal
External Entity
User
gets information
leads to
leads to
leads to
gets information
Model users
Ontology based information source
gets information
Filter information
Deliver recommendations
achieves
achieves
achieves
achieves
achieves
achieves
achieves
achieves
achieves
achieves
Implicit profile acquisition
Explicit profile acquisition
Ontology instance user model creation and update
Grouping of user models
Information items based on Ontology instance representation
Similarity analysis
Personalized recommendation production
Delivery of personalized recommendation

Figure 8. The ONTOSERS Goal Model

Figure 9 shows the variants of the specific goal "Model users" in the ONTOSERS domain model. The “Model Users” specific goal has a variation point with groups of responsibilities for user profile acquisition, being possible to choose between three alternative variants: “Implicit Profile Acquisition”, “Explicit Profile Acquisition” or both.

The role modeling task associates the responsibilities, either common or variants, identified in the goal modeling task to the roles that will be in charge of them. The pre and post-conditions that must be satisfied before and after the execution of a responsibility are also identified. Finally, the knowledge required from other entities (roles or stakeholders) for the execution of responsibilities and the knowledge produced from their execution is identified. This task produces a set of role models, one for each specific goal or, having it one or more variation points, one role model for each variant, specifying roles, responsibilities, pre- and post-conditions, knowledge and relationships between these concepts.

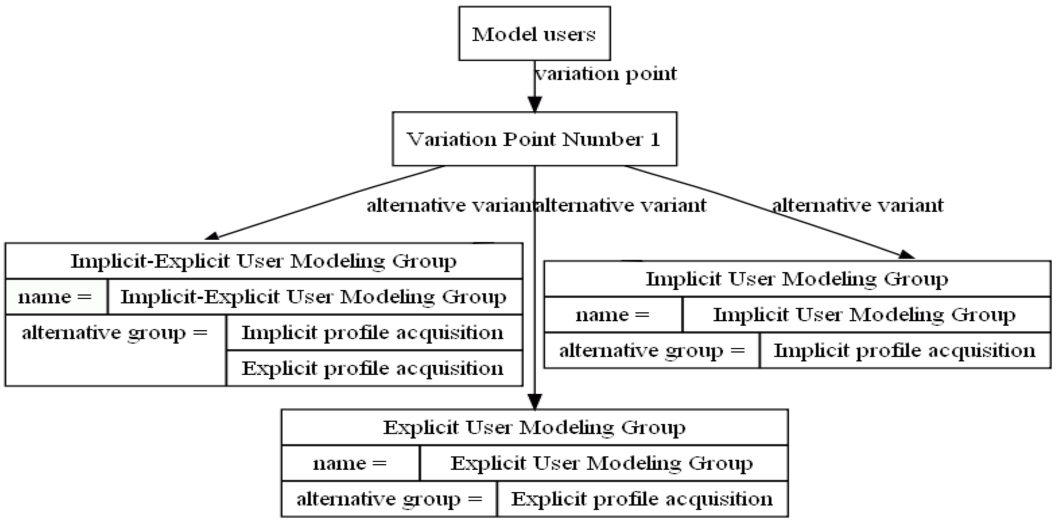

Model users
variation point
Variation Point Number 1
alternative variant
alternative variant
alternative variant
Implicit-Explicit User Modeling Group
name = Implicit-Explicit User Modeling Group
alternative group = Implicit profile acquisition
Explicit profile acquisition
Implicit User Modeling Group
name = Implicit User Modeling Group
alternative group = Implicit profile acquisition
Explicit User Modeling Group
name = Explicit User Modeling Group
alternative group = Explicit profile acquisition

Figure 9. The variation point of the specific goal "Model users" and the alternative groups of responsibilities variants

Figure 10 shows some role variants in the role models of the ONTOSERS domain model produced through variability modeling of the role modeling task. For each group of responsibilities in an alternative variant of Figure 9 a role model is developed. For instance, Figure 10 shows the semantic relationships relating the role “User Monitor”, derived from the “Implicit profile acquisition” responsibility and from the groups of alternative responsibilities in the “Implicit User Modeling Group” and “Implicit-Explicit User Modeling Group” variants of Figure 9 to the alternative role models “Implicit User Modeling ONTOSERS Role Model” and “Implicit-Explicit User Modeling ONTOSERS Role Model”.

Figure 11 shows an example of a variant role interaction model of the ONTOSERS domain model produced through variability modeling of the role interactions modeling task. For each alternative variant in Figure 9 a role interaction model is developed. Figure 11 shows the role interaction model with the interactions between roles and stakeholders needed to accomplish the ‘Model user” specific goal under the variant “Implicit-Explicit User Modeling Group” of Figure 9. The Monitor role captures user navigational behavior. A user profile, acquired implicitly, is transferred to the “User Modeler” role so that it can create a user model. Another alternative is explicit profile acquisition in which the user explicitly specifies his/her interests through the “Input Interface” role that sends the profile to the “User Modeler” role.

Finally, a reusable user interface prototype is developed by identifying the interactions of users with the system family.

3.2. The Application Requirements Engineering Tasks

In this phase, reuse of domain models is supported by the ONTORMAS tool. In ONTORMAS, the selection of software artifacts is supported by semantic retrieval, where the user inputs a query specifying the product features he/she intends to reuse and gets from the repository the available artifacts satisfying his/her query. After the selection of the artifact that most closely matches their needs, users should check if the artifact can be integrally reused or if it needs adaptations and/or integrations with other artifacts.

The concepts modeling task aims at performing a brainstorming of the application concepts and their relationships, representing them in a concept model.

The purpose of the goal modeling task is to identify the goals of the application, the stakeholders with which it cooperates and the responsibilities needed to achieve them. Its product is a goal model, specifying the general and specific goals of the application along with the stakeholders and responsibilities. This task should be reuse-intensive. From the concept model and from a first draft of the goal model, possible terms for searching and reusing goals in already available domain models can be revealed.

If a general goal is identified, the corresponding goal model in a domain model is selected for reuse. If a specific goal is identified, this goal, sub-goals in a possible hierarchy, related responsibilities and stakeholders in a goal model of a domain model are selected for reuse. Otherwise, the goal model is constructed from scratch.

If a selected specific goal or sub-goals in its hierarchy have associated variation points, they should be analyzed to select and possible reuse the appropriate variants of alternative or optional groups of responsibilities by considering both functional and non-functional requirements of the specific application. Only one group of responsibilities in an alternative variant can be selected for reuse. Zero or more groups of responsibilities in an optional variant can be selected for reuse.

Figure 13 illustrates the goal model of InfoTrib. To construct it, first, a semantic search in the ONTORMAS knowledge base with the term “recommendation” was done (Figure 12). The general goal “Provide recommendations using semantic web technologies” was retrieved through the search. Therefore, the corresponding goal model was selected for reuse, in this case, the goal model of ONTOSERS (Figure 8), part of the ONTOSERS domain model. From the variation point of the “Model users” specific goal (Figure 9), the ”Explicit profile acquisition” responsibility variant was selected in order to support just the functional requirement of explicit acquisition of user profiles.

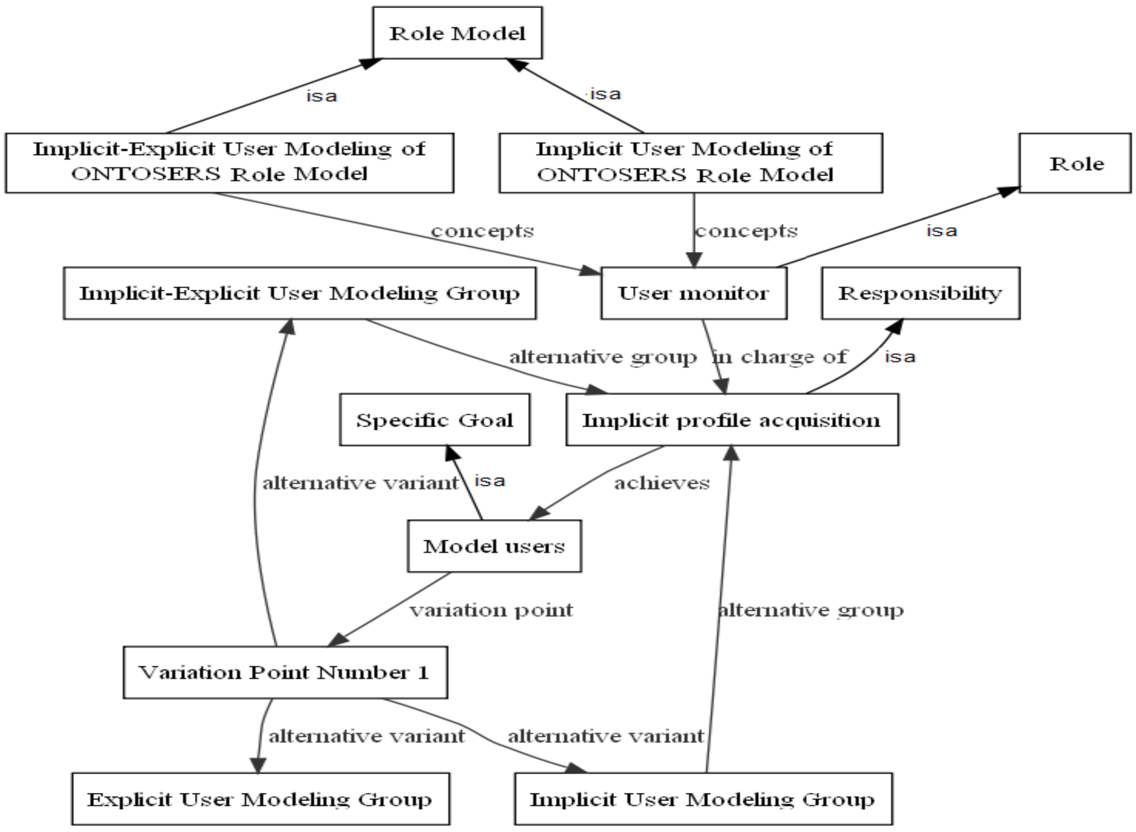
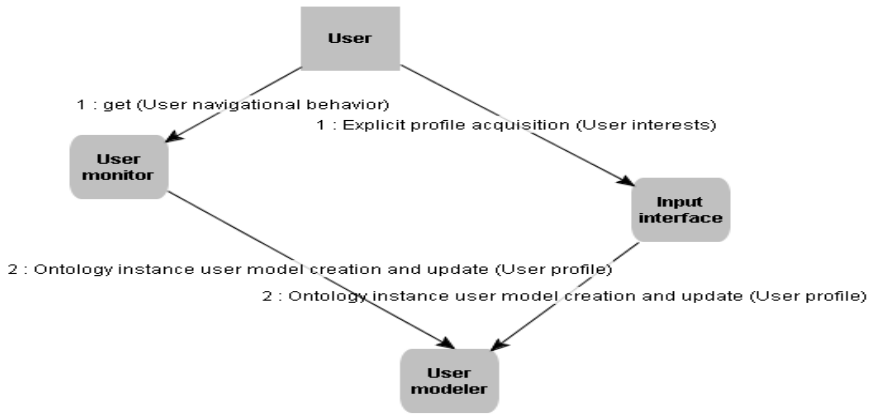


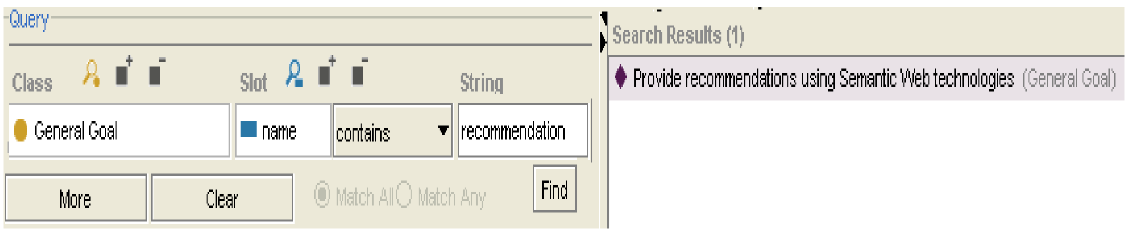
Figure 10. Variability modeling of the role modeling task: the variant roles in the alternative role models



User
1 : get (User navigational behavior)
1 : Explicit profile acquisition (User interests)
User monitor
Input interface
2 : Ontology instance user model creation and update (User profile)
2 : Ontology instance user model creation and update (User profile)
User modeler

Figure 11. Role Interaction Model of the "Model Users" Specific Goal under the variant "Implicit-Explicit User Modeling Group"

From the variation point of the "Filter Information" specific goal, the "Information Items based on Ontology Instance Representation" responsibility variant was selected for providing content-based information filtering. The name of the external entity "Ontology based information source" was specialized to "ONTOTRIB", the ontology that defines the Tax Law concepts and relationships.



General Goal | name | contains | recommendation

Provide recommendations using Semantic Web technologies (General Goal)

Figure 12. A simple query for general goals in ONTORMAS

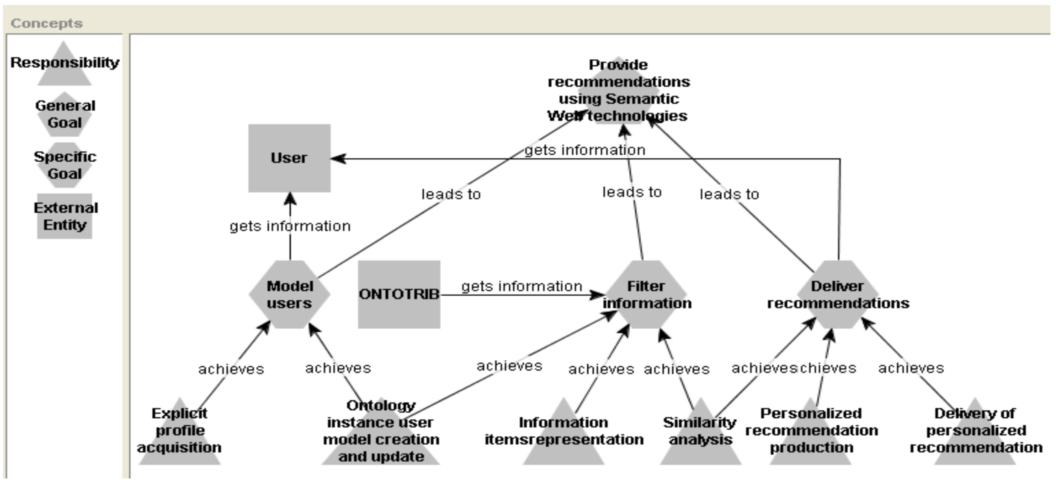


Figure 13. The Goal Model of INFOTRIB

The role modeling task associates the responsibilities identified in the goal modeling task to the roles that will be in charge of them. The pre and post-conditions that must be satisfied before and after the execution of a responsibility are also identified. Finally, the knowledge required from other entities (roles or stakeholders) for the execution of responsibilities and the knowledge produced from their execution is identified. A set of role models, one for each specific goal in the goal model is constructed in this task, with or without reuse.

The following rules apply for the reuse activities performed during this modeling task:

- If a similar general goal is identified during the goal modeling task, thus reusing fully or partially a goal model then, the set of role models, already available in the corresponding domain model and associated to each reused specific goal, will be reused and eventually adapted for the previously customized specific goals and selected responsibilities from groups of alternative or optional variants.
- Otherwise, if a set of similar specific goals are identified during the goal modeling task, thus reusing partially a goal model, then the set of role models already available in the corresponding domain model, associated with the similar specific goal will be reused and eventually adapted, considering selected responsibilities from groups of alternative or optional variants.
- Otherwise, if the goal model is constructed from scratch, then the set of role models will be also constructed from scratch, one for each specific goal.

Please note that, in this task, reuse is implicitly supported by the semantic relationships that associate a specific goal with a role model.

For instance, in the example of Figure 13, a similar general goal was identified during the goal modeling task, thus reusing partially a goal model, having the “Explicit profile acquisition” responsibility variant associated to the “Model users” specific goal variation point and the “Information Items based on Ontology Instance Representation” responsibility variant associated to the “Filter Information” specific goal variation point. Then, the set of role models, already available in the ONTOSERS domain model and associated to each reused specific goal and selected variants will be reused.

The role interaction modeling task aims at identifying how external and internal entities should cooperate to achieve a specific goal. For that, responsibilities of roles are analyzed along with their required and produced knowledge specified in a role model. A set of role interaction models is reused in this modeling task, one for each specific goal. The interactions are numbered according to their sequencing. Similar rules to the ones of the role modeling task apply for the reuse activities performed during this modeling task.

For the construction of the user interface prototype of the specific application, the generic interfaces associated to a reused external entity are selected and customized according to the specific goal with which it is related.

4. Related Work

Several approaches for agent-oriented software development, like GAIA [Zambonelli, F., Jennings, N. and Wooldridge, M. 2003], PASSI [Cossentino, M. et al. 2004] and TROPOS [Bresciani, P. et al. 2004] and some domain engineering processes [Nunes, I. et al. 2009], have been already developed to increase the productivity of the software development process, the reusability of generated products, and the effectiveness of project management.

GAIA is a methodology based in human organization concepts. It supports the analysis and design phases of multi-agent system development. Tropos is an agent-oriented software development methodology supporting the complete multi-agent development process. It is based on the i* organizational modeling framework. PASSI is a process for multi-agent development integrating concepts from object-oriented software engineering and artificial intelligence approaches. It allows the development of multi-agents systems for special purposes as mobiles and robotics agents and uses an UML-based notation. In [Nunes, I. et al. 2009] is described a domain engineering process which focuses on system families including domain scoping and variability modeling. This process integrates a product line UML-based method, the PASSI methodology and a modeling language for developing multi-agent system product lines.

Table 1 summarizes and compares some characteristics of GAIA, PASSI, TROPOS, MADAE-Pro and the domain engineering process described above. All the approaches propose an iterative life cycle, where a software product follows several refinements during the development process. With the exception of GAIA, in all other approaches the life cycle is also incremental, where a software product is represented in several models to facilitate its understanding.

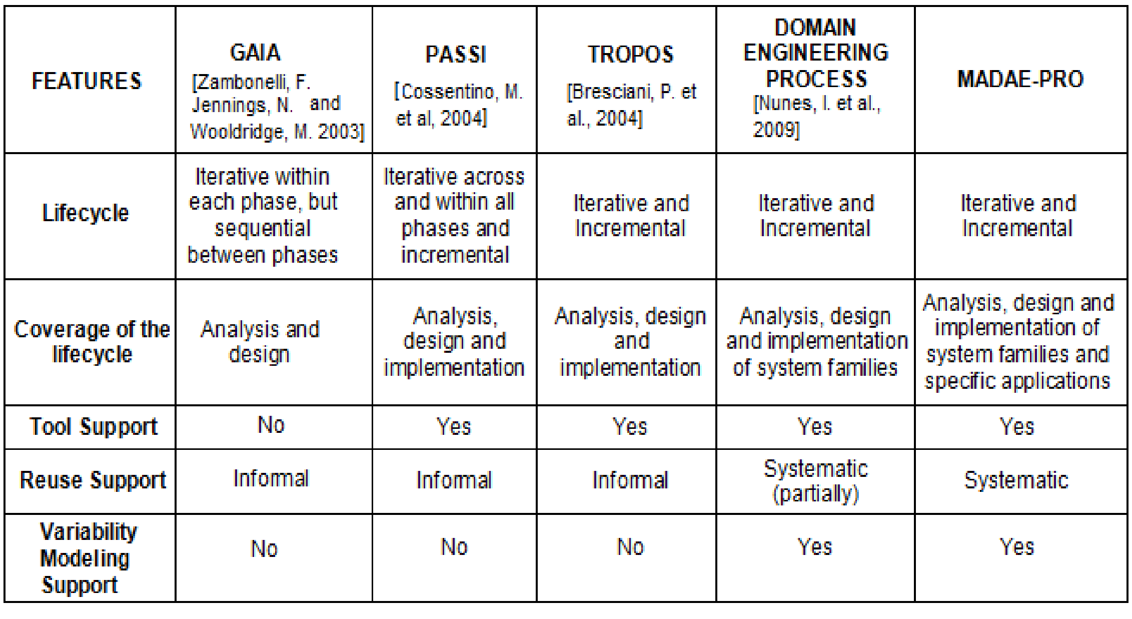
For the supported development phases, all these approaches cover analysis and design while PASSI, TROPOS and MADAE-Pro also support the implementation phase. The domain engineering process described above covers the domain engineering phases of early and late requirements, domain design and domain realization. To our knowledge, only MADAE-Pro provides support for both domain and application engineering.

For the available development tools, PASSI is supported by PTK, a Rational Rose plug-in allowing modeling in AUML and code generation. The application of TROPOS is assisted by the TAOM-Tool [Perini, A. and Susi, A. 2004], an Eclipse plug-in allowing system modeling with the i* framework. The MADAE-Pro process is supported by the ONTORMAS tool that allows the modeling and storage of individual applications and families of multi-agent applications as instances of the ONTORMAS ontology. GAIA does not report a tool support yet.

For reuse activities, GAIA and TROPOS allow the reuse of models and code in an informal way. PASSI permits the reuse of source code from class and activity diagrams. The domain engineering process described in [Nunes, I. et al. 2009] is based on the concept of “feature”, a system property relevant to some stakeholder, used to capture commonalities and to discriminate products in software product lines. However, this process does not offer guidelines for the selection, adaptation and integration of software artifacts. MADAE-Pro process allows reuse of both models and source code of software products giving support for their selection, adaptation and integration.

For the variability modeling support, only MADAE-Pro and the domain engineering process described in [Nunes, I. et al. 2009] support it. This approach uses an extension of UML for modeling variabilities [Goma, H. 2005] while MADAE-Pro uses MADAE-ML, an ontology-driven modeling language.

Table 1. A comparison of agent-oriented software development approaches



FEATURES	GAIA [Zambonelli, F. Jennings, N. and Wooldridge, M. 2003]	PASSI [Cossentino, M. et al, 2004]	TROPOS [Bresciani, P. et al., 2004]	DOMAIN ENGINEERING PROCESS [Nunes, I. et al., 2009]	MADAE-PRO
Lifecycle	Iterative within each phase, but sequential between phases	Iterative across and within all phases and incremental	Iterative and Incremental	Iterative and Incremental	Iterative and Incremental
Coverage of the lifecycle	Analysis and design	Analysis, design and implementation	Analysis, design and implementation	Analysis, design and implementation of system families	Analysis, design and implementation of system families and specific applications
Tool Support	No	Yes	Yes	Yes	Yes
Reuse Support	Informal	Informal	Informal	Systematic (partially)	Systematic
Variability Modeling Support	No	No	No	Yes	Yes

Two main features distinguish MADAE-Pro from other existing approaches. First, it provides support for reuse in multi-agent software development, through the integration of the concepts of Domain Engineering and Application Engineering. Second, it is a knowledge-based process where models of agents and frameworks are represented as instances of the ONTORMAS ontology. Thus, concepts are semantically related allowing effective searches and inferences thus facilitating the understanding and reuse of models during the development of specific applications in a domain. Also, the ontology-driven models of MADAE-Pro can be easily documented, adapted and integrated.

5. Conclusion and Further Work

This work described MADAE-Pro, a knowledge-based process model for Multi-agent Domain and Application Engineering, emphasizing the description of its domain analysis and application requirements engineering phases, showing how software artifacts produced on the first phase are reused in the last one.

The SPEM process modeling language has been used to formalize the process, thus providing a standard, documented and ambiguity free representation of MADAE-Pro. The formalization of MADAE-Pro has allowed the systematic application of its life cycle along with the MADEM and MAAEM methodologies for the construction of multi-agent system families and specific multi-agent applications as well. Also, this formal model provides a basic framework for automating the MADAE-Pro development tasks.

The ONTORMAS tool helps developers on the systematic application of the MADAE-Pro process. The software artifacts produced through its modeling tasks are instantiated in the ONTORMAS knowledge base, which it is used as a repository of reusable software artifacts. The semantic representation of software products increase reuse effectiveness, providing more precision on software retrieval.

MADAE-Pro has been evaluated with case studies approaching both the development of application families [Girardi, R. and Marinho, L.2007] [Mariano, R. at al. 2008] and specific applications [Drumond, L. and Girardi, R. 2008] [Nunes, I. et al. 2009][Newton, E. and Girardi, R. 2007]. The process proved to be suitable for the identification and representation of the fixed and variable parts of software abstractions of the ONTOSERS family, thus making possible its reuse on the development of specific applications [Mariano, R. 2008].

MADAE-Pro is part of a project for the improvement of multi-agent development techniques, methodologies and tools. With the knowledge base provided by ONTORMAS, an expert system is being developed, aiming at automating various tasks of both MADEM and MAAEM, thus allowing fast application development and partial code generation.

MADAE-Pro currently supports compositional reuse, based on the selection, adaptation and composition of software artifacts. A generative approach for reuse has been explored with the specification of the GENMADEM methodology and the ONTOGENMADEM tool [Jansen, M. and Girardi, R. 2006]. ONTOGENMADEM provides support for the creation of Domain Specific Languages to be used on the generation of a family of applications in a domain. Further work will extend ONTORMAS for supporting ONTOGENMADEM allowing generative reuse in Multi-agent Application Engineering.´

Also, to evaluate MADAE-Pro, it is being planned the development of new application families and specific applications in other domains of interest.

References

Arango, G. (1988). "Domain Engineering for Software Reuse". Ph.D. Thesis. Department of Information and Computer Science, University of California, Irvine.

Bellifemine, F., Caire, G. Poggi A. and Rimassa, G. (2003) "JADE A White Paper". Exp v. 3 n. 3, Sept., http://jade.tilab.com/

Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J. and Perini, A. (2004) "TROPOS: An Agent-Oriented Software Development Methodology", In: Journal of Autonomous Agents and Multi-Agent Systems, Kluwer Academic Publishers Volume 8, Issue 3, pp. 203-236.

Cossentino, M., Sabatucci, L., Sorace, S. and Chella, A. (2004) "Patterns reuse in the PASSI methodology". In: Proceedings of the Fourth International Workshop Engineering Societies in the Agents World (ESAW'03), Imperial College London, UK, pp. 29-31.

Czarnecki, K., Eisenecker, U. W. (2000) "Generative Programming: Methods, Tools, and Applications". ACM Press/Addison-Wesley Publishing Co., New York, NY, 2000.

Dileo, J., Jacobs, T. and Deloach, S. (2002) "Integrating Ontologies into Multi-Agent Systems Engineering". Proceedings of 4th International Bi-Conference Workshop on Agent Oriented Information Systems (AOIS 2002), (pp. 15-16). Bologna (Italy).

Drumond, L. and Girardi, R. (2008) "A Multi-agent Legal Recommender System". Artificial Intelligence and Law (Dordrecht), March, pp. 175-207.

Friedman-Hill, E. (2003). "JESS in Action". Manning Publications.

Girardi, R. (1992) "Application Engineering: Putting Reuse to Work". In: Dennis Tsichritzis (ed.). (Org.). Object Frameworks. Geneve: CUI, v. I, Université de Geneve, p. 137-149.

Girardi, R. and Marinho, L.(2007) "A Domain Model of Web Recommender Systems based on Usage Mining and Collaborative Filtering", Requirements Engineering Journal, vol.12 n. 1, Ed. Springer-Verlag, p. 23-40.

Gennari, J. et al. (2002). "The evolution of Protégé: An environment for knowledge-based systems development", Technical Report SMI-2002-0943.

Girardi, R., Leite, A. (2008). "A Knowledge-based Tool for Multi-Agent Domain Engineering". Knowledge-Based Systems, April, p. 604-611.

Goma, H. (2005). "Designing Software Product Lines with UML: From Use Cases to pattern-based Software Architectures", Addison-Wesley Object Technology Series.

Gruber, T. R. (1995) “Toward Principles for the Design of Ontologies used for Knowledge Sharing, International Journal of Human-Computer Studies”. Nº 43, 1995, pp. 907-928.

Harsu, M. (2002) “A Survey of Domain Engineering”. Report 31, Institute of Software Systems, Tampere University of Technology, December.

Jansen, M. and Girardi, R. (2006) “GENMADEM: A Methodology for Generative Multi-agent Domain Engineering”. In: Proceedings of the 9th International Conference on Software Reuse, Torino. Lecture Notes in Computer Science (LNCS), v. 4039. Berlin: Springer-Verlag, p. 399-402.

Leite, A., Girardi, R. and Cavalcante, U. (2008a) “An Ontology for Multi-Agent Domain and Application Engineering”. In: Proceedings of the 2008 IEEE International Conference on Information Reuse and Integration (IEEE IRI-08), Las Vegas, 2008, pp. 98-103.

Leite, A., Girardi, R. and Cavalcante, U. (2008b) “MAAEM: A Multi-agent Application Engineering Methodology”. In: Proceedings of the 20th International Conference on Software Engineering and Knowledge Engineering, Redwood City. Knowledge Systems Institute, pp. 735-740.

Mariano, R., Girardi, R., Leite, A., L. Drumond and D. Maranhão, A. (2008) “Case Study on Domain Analysis of Semantic Web Multi-agent Recommender Systems”. In: Proceedings 3th International Conference on Software and Data Technologies, Porto. Portugal, p. 160-167.

Mariano, R. (2008) “Development of a Family of Recommender Systems based on the Semantic Web Technology and its Reuse on the Recommendation of Legal Tax Information Items”, Master Dissertation, Federal University of Maranhão, São Luís. (In Portuguese)

Newton, E. and Girardi, R. (2007) “PROPOST: A knowledge-based tool for supporting Project Portfolio Management”. In: International Conference on Systems Engineering and Modeling - ICSEM'07, Haifa. Proceedings of ICSEM'07, p. 98-103.

Nunes, I., Kulesza, U., Nunes, C. and Lucena, C. A. (2009) “A Domain Engineering Process for Developing Multi-Agent Systems Product Lines”. Proceedings of the 8th International Conference of Autonomous Agents and Multi-agent Systems (AAMAS 2009), Budapest, pp. 10-15.

Odell, J., Parunak, H. V. D. and Bauer, B. (2000) “Extending UML for Agents. Proc. of the Agent-Oriented”, Information Systems Workshop at the 17th National Conference on Artificial Intelligence, pp. 3-17.

Perini, A. and Susi, A. (2004) “Developing Tools for Agent-Oriented Visual Modeling. In G. Lindemann”, Multi-agent System Technologies, Proceedings of the Second German Conference, MATES 2004, number 3187 in LNAI, Springer-Verlag, pp. 169–182.

Pohl, K., Bockle, G. and Linden, F. (2005) “Software Product Line Engineering: Foundations, Principles and Techniques”, Springer Verlag, USA.

SPEM - Software Process Engineering Metamodel Specification, Available at: http://www.omg.org/cgi-bin/doc?formal/02-11-14.pdf, Last access, Dez., 2010.

Shadbolt, N., Hall, W., Berners-Lee, T. (2006) “The semantic web revisited”. In:

Intelligent Systems, vol. 21, n. 3, pp. 96–101.

Zambonelli, F., Jennings, N. and Wooldridge, M. (2003) “Developing multi-agent systems: The Gaia methodology”. ACM Transactions on Software Engineering and Methodology, pp. 417-470.