A Methodology to Engineer Graded BDI Agents

Ana Casali
Depto. de Sistemas e Informática FCEIA - UNR
Av Pellegrini 250, 2000 Rosario, Argentina.
acasali@fceia.unr.edu.ar

and

Lluís Godo and Carles Sierra
Institut d'Investigació en Intel·ligència Artificial (IIIA) - CSIC
Campus UAB, 08193 Bellaterra, Catalunya, España.
godo, sierra@iiia.csic.es

Abstract

In this work we present a methodological framework to engineer graded BDI agent-based systems. The graded BDI agent model allows to specify an agent's architecture able to deal with the environment uncertainty and with graded mental attitudes. We work up previous approaches on software engineering process, adapting and extending them, in order to develop agents with a more complex internal architecture.

Keywords: Agent-Based Software Engineering, Graded BDI Agents.

1 INTRODUCTION

Agent technology has received a great deal of attention in the last few years and, as a result, many software applications are developed using this technology. In spite of the different developed agent theories, languages, architectures and the successful agent-based applications, further work is needed for specifying (and applying) techniques to develop applications using agent technology. The role of agent-oriented methodologies is to assist in all the phases of the life cycle of an agent-based application, including its management.

Many different Agent Oriented Software Engineering (AOSE) approaches have been proposed, a survey of some of them can be seen in [1]. Each of the methodologies has different strengths and weaknesses, and diverse specialized features to support different aspects of their intended application domains. Most of the methodologies have shown that there is a conceptual level for analysing the agent-based systems, no matter the agent theory, agent architecture or agent language they are supported by. This conceptual level should describe fundamentally the external view point of agents by the Agent Models (the characteristics/tasks of each agent) and the Society Models (the relationships and interactions between the agents). We consider that is important for a methodology also to include the agent development from an internal point of view, selecting the necessary models to develop its architecture, as is pointed in [7].

Among the architectures proposed to give the agent-based systems a formal support, a relevant approach is the BDI architecture proposed by Rao and Georgeff [8]. This model is based on the explicit representation of the agent's beliefs (B), desires (D) and intentions (I). Indeed, this architecture has evolved over time and it has been applied, to some extent, in several of the most significant multiagent applications developed up to now.

With the purpose of making the BDI architecture more flexible, we have proposed a general model for Graded BDI (g-BDI) Agents. This model allows to specify an agent's architecture able to deal with the environment uncertainty and with graded mental attitudes, see [3] for first results. In this model, belief degrees represent to what extent the agent believes a formula is true. Degrees of positive or negative desires allow the agent to set different levels of preference or rejection respectively. Intention degrees give also a preference measure but, in this case, modeling the cost/benefit trade-off of reaching an agent's goal. Then, agents having different kinds of behavior can be modeled on the basis of the representation and interaction of these three attitudes. The graded BDI model we have developed is based in the notion of *multi-context system* (MCS) used to design complex logical systems and particularly, agent systems [9]. This framework allows the definition of different formal components and their interrelation.

Since there is no standard agent architecture, the design of the agents needs to be customised to each agent architecture. We want to set a methodology in order to engineer graded BDI agent systems.

Software Engineering Process for BDI Agent Based Systems

There are few works on Software Engineering Process for BDI Agent Based Systems. Kinny et al. in [7] proposed a methodology for agent-oriented analysis and design, focusing upon the BDI model of agents. In specifying an agent system, they have found that is highly desirable to adopt specialized set of models which operate at two distinct levels of abstraction. First, from the external viewpoint, the system is decomposed into agents, modeled as complex objects characterized by their purpose, responsibilities and services they perform, the information they require and maintain, and their external interactions. Second, from the internal point of view, the elements required by a particular agent architecture must be modeled for each agent. More recently, Jo et al. in [5] proposed the BDI Agent Software Development Process (BDI-ASDP) as a specialization of traditional and Object Oriented software engineering methodologies, embracing several steps. A similar approach of software engineering process is presented by Zhang et al. in [11].

All these proposals share the same principal stages. They take advantage of different artifacts proved to be useful in Object-Oriented Software Engineering, adapting them to their purpose. They are designed to identify the beliefs, desires, and intentions for agents during the software analysis and design phases. Following the natural style of human thinking "goal-plan-data", their approaches first extract the desires from the requirement, and then create the proper plans. Finally they find the beliefs. The task of agent recognition is done during the BDI process after the goals and plans identification.

Despite of having common issues, we can remark that in the work of Jo et al. [5] and of Zhang et al. [11], there is not a clear separation in the agent analysis and design from an external and internal viewpoint, as is pointed in [7]. More specifically, their proposals contain the following stages:
(1) They use some artifacts to specify system requirements (such as External Use Cases) and to extract goals (desires) from them.
(2) They use Dynamic Models (such as Internal Use Cases, Sequence Diagrams, and Activity Diagrams) to provide a more precise description of each goal and its corresponding plan (intentions).
(3) Then, a role analysis is performed from the list of goals and their corresponding plans. The relevant roles and their interactions (role composition) are taken into account to define the set of agents.
(4) Finally, using Data Models (such as Data Flow Diagrams) they propose to obtain the environment information (beliefs) that is necessary for the goals completion.
This software modeling is processed and modified iteratively. After a complete BDI specification has

been described, then it is assigned to an agent.

In this work we present a methodological framework based on these previous works, to engineer graded BDI agents. We work up these previous approaches, adapting and extending them, in order to engineer agents with a more complex internal architecture. In this sense, we make more emphasis in the separation of the agent design process from an external an internal point of view, adding some steps for the internal design of the agents. Particularly, we focus on a graded BDI model of agent and we present an insight of the process of its multicontext specification.

This paper is organized as follows. In Section 2, the Graded BDI agent model is briefly revised. In Section 3, we outline the development process of g-BDI Agent-Based Systems. Following, in Section 4 the most important stages of the process are described and a case study is used to illustrate them. Finally, in Section 5 we present some conclusions and future work.

2 GRADED BDI AGENT MODEL

The architecture proposed is inspired by previous work about multi-context agent's specification [9]. The MCS specification contains three basic components: units or contexts, logics, and bridge rules, which channel the propagation of consequences among theories. Thus, an agent is defined as a group of interconnected units: $\langle \{C_i\}_{i \in I}, \Delta_{br} \rangle$, where each context $C_i \in \{C_i\}_{i \in I}$ is the tuple $C_i = \langle L_i, A_i, \Delta_i \rangle$ where L_i, A_i and Δ_i are the language, axioms, and inference rules respectively. When a theory $T_i \in L_i$ is associated with each unit, the specification of a particular agent is complete. The deduction mechanism of these systems is based on two kinds of inference rules, internal rules Δ_i, and bridge rules Δ_{br}, which allow to embed formulae into a context whenever the conditions of the bridge rule are satisfied. In our model, we have *mental* contexts to represent beliefs (BC), desires (DC), intentions (IC), and a social context (SC) which represents the trust in other agents. We also consider two *functional* contexts: for Planning (PC) and Communication (CC). In summary, the BDI agent model is defined as a tuple $A_g = (\{BC, DC, IC, PC, CC\}, \Delta_{br})$. The overall behavior of the system will be the result of the logical representation of each intentional notion in the different contexts and their interaction by means of the bridge rules.

In order to represent and reason about graded notions of belief, desire and intention, we use a modal many-valued approach. In particular, we shall follow the approach where uncertainty reasoning is dealt with by defining suitable modal theories over suitable many-valued logics. For instance, let us consider a Belief context where belief degrees are to be modeled as probabilities. Then, for each classical formula φ, we consider a modal formula $B\varphi$ which is interpreted as "φ is probable". This modal formula $B\varphi$ is then a *fuzzy* formula which may be more or less true, depending on the probability of φ. In particular, we can take as truth-value of $B\varphi$ precisely the probability of φ. Moreover, using Łukasiewicz infinitely-valued logic, we can express the governing axioms of probability theory as logical axioms involving modal formulae. To set up an adequate axiomatization for our belief context logic we need to combine axioms for the crisp formulae, axioms of Łukasiewicz logic for modal formulae, and additional axioms for B-modal formulae according to the probabilistic semantics of the B operator. The same modal many-valued logic approach is used to represent and reason under graded attitudes in the other mental contexts. The formalization of the adequate logics for the different contexts are described in [3, 4].

3 THE DEVELOPMENT PROCESS OF G-BDI AGENT-BASED SYSTEMS

In our approach we need first to recognize the agents composing a multiagent system. Then, in order to model BDI agents, we must discover from the system requirements which are the different mental attitudes. In some cases, the inclusion of additional mental attitudes may be important, but in our current methodological guide we consider fundamentally these three basic attitudes (i.e. BDIs).

We first find the goals (positive desires) from the system requirements. In this stage we take a bipolar view of desires, allowing to distinguish the goals from rejected states (negative desires). Also, we consider important to incorporate degrees on desires to set different levels of preference or rejection, respectively. Starting from the system goals or positive desires, responsibilities are defined as an intermediary step in the role identification process (i.e. a responsibility has a list of goals attached to it and a role has a list of responsibilities). The agents in the multiagent system will be defined through the integration of the relevant role models. The role composition analysis is needed because when the agent carries out their responsibilities and interacts/collaborates, there may be synergy between the different roles played by the agent [11]. Following with the design process, the model of each agent (a BDI model) must be analyzed, for which we propose a methodology adapting some steps of the BDI process presented in [5, 11] and adding other stages we consider important for our purpose.

To design a particular agent in the multiagent system, we restrict the system goals to the ones assigned to this agent. Beginning with these goals, we can set the agent's positive and negative desires. The second stage is to determine how the agent will make some plans, to fulfill the positive desires and to avoid the negative ones. These will constitute the feasible plans for the agent. In order to execute these plans a set of beliefs is needed, describing the inner state of the agent and the state of the world. Sometimes these beliefs may be uncertain, imprecise or incomplete. Finally, we must establish what factors the agent will take into account to find the intention to follow. In our approach, the notion of intention is related to a pair desire/plan, where the desire is the one that the agent will try to satisfy by executing the plan. The agent may have different sets of positive desires, that may be cooperative or competitive, and in order to satisfy them there may be alternative plans, in turn with associated costs. Then, in our approach the agent will consider pairs desire/plan with the best cost/benefit relation for reaching a given goal by executing a feasible plan. As a result of this analysis the agent has to decide which intention (a chosen goal) to follow by executing the best plan towards it. In an agent design, this deliberation process and the elements involved must be both specified.

Following the flow "goals-feasible plans-beliefs-intention", our approach will first extract the desires from the requirements, then it will analyze the possible plans towards them and the beliefs involved, and finally it will set its intentions by modeling an appropriate deliberation process. In our approach we must also find the interactions between the BDI attitudes and the kind of information involved in each one.

The development process for BDI agent-based systems is depicted in Figure 1. The white boxes represent some artifacts or tools that give support to the principal steps in this process, some of them are described in Section 4. The colored boxes are the necesary parts of the internal agent's design. As our approach is focused on the development of graded BDI agents, this internal blocks are the mental attitudes (i.e. BDIs) and the bridge rules, relating them. The arrows intend to show a possible sequence between the different steps. Since in practice the methodology is iterative, analysts or designers may freely move between steps and phases and each successive iteration will produce additional details to finally provide a complete, yet consistent system design.

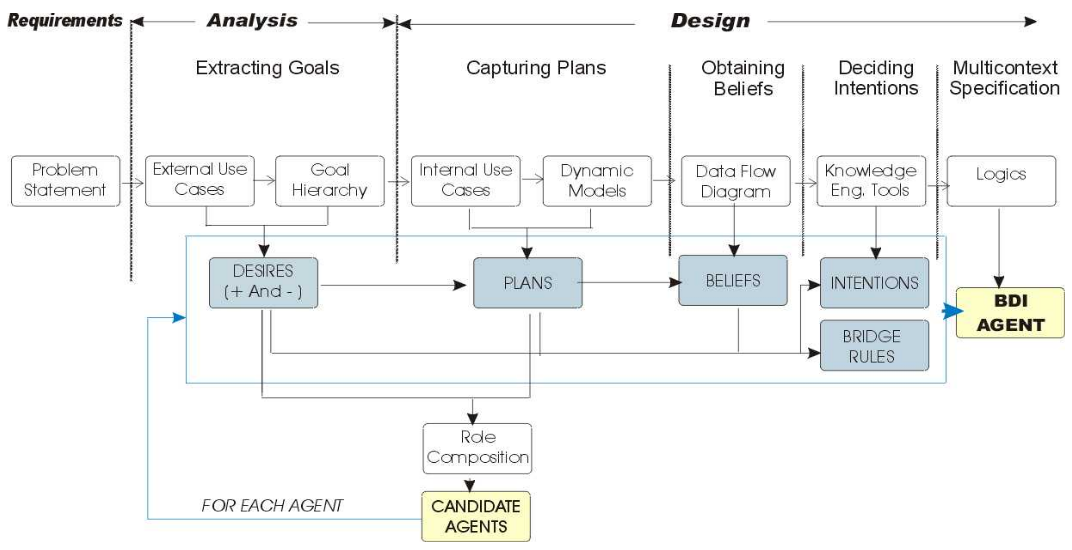

Figure 1: Development Process for BDI Agent-Based Systems

4 DEVELOPMENT STAGES: A CASE STUDY

In the following subsections we describe the most important stages and steps of the software engineering process presented in the previous Section. To illustrate and clarify these different steps, we describe the process using a Case Study in the Tourism Domain.

4.1 Requirement Analysis

As usual in Software Engineering, the requirement analysis is the initial part of the software development Process. It will assist us to understand the purpose of the system and how to construct it.

Step 1: Initial Problem Statement

This is the previous and fundamental step for the System Analysis, where the problem that the system is expected to solve is described. It is a hight level conceptualization of the system from the user's point of view, and describes the services that the system will provide. It is the input to capture the system goals.

Case Study:

We want to design a Travel Assistant System, a recommender system on Argentinian touristic plans. This system will be in charge of looking for different holiday plans in Argentinian destinations, in order to satisfy the desires of a tourist. The customer's desires may be preferences about geographic conditions, infrastructures, activities, travelling forms, accomodation, etc. He/she may also have different rejections or restrictions, as for example a given maximum amount he can spend. The touristic plan the system is expected to offer must be the best choice among the packages supplied by a set of tourism operators. The system has to decide which touristic package (plan) to recommend taking into account the interests of the customer, the expected satisfaction of the preferences by the plan, its cost and the trust in the plan supplier.

4.2 System Analysis: Extracting Goals

During this phase, the investigation on the problem and its requirement is deepened. We focus on finding the multiagent system goals, which in turn will result in the first element of agent's specification: the desires.

Step 2: External Use Cases

The External Use Case treat the system as a black box, and show how the entities outside of the system interact with it. During the External Use Case process, we identify the services of the system being developed from an external point of view; we do not describe the internal working components or design of the system. The External Use Case captures who (actor) does what (interaction) with the system, for what purpose (goal), without dealing with the system internals. In an iterative process, more detailed use cases are given.

Case Study:

In order to discover all the functions that the Recommender system should provide, we use external use cases. See the next example:

Name: Look for Feasible Tourist Packages
Actors: tourist, tourism-operators (touristic plan reservories)
Description:
- A Tourist wants a recommendation of touristic plan in Argentinien destinations.
- He/she has different preferences (geography, transport, activities, acommodation, etc)
- He/she may have some rejections or restrictions (about zones, distances, acommodation, etc)
- The system has touristic packages provided by different Tourism Operators.
- The system wants to find feasible plans satisfying most of the Tourist's preferences and avoiding the rejections.

Preconditions: The system is asked for a recomendation and a list of preferences/rejections are given.
Sucess/Postcondition: The system finds a set of feasible packages.
Extensions: System fails
- the system finds no plan satisfying both a set of preferences and the rejections conditions.
- ask the customer if some of the rejections can be removed.
- ask the providers for more alternatives satisfying the list of preferences

Step 3: Goal Hierarchy

From the previous stages a set of goals (positive desires) can be structured considering possible inter-relationships between them. A Goal Hierarchy Diagram should be used to represent these relationships, beginning with the overall system goal. Each goal may be different, not only in the hierarchy (w.r.t. a global goal) but also in importance. For dealing with this, we propose to use graded expressions (in $[0, 1]$) in order to represent the different levels of importance of desires at the same level of the hierarchy. In this step, we also propose to distinguish another set of graded rejected states (negative desires), representing the situations that we do not want the system reach, also with different importance degrees.

Case Study:

The overall goal of the system is to satisfy the tourist with the best touristic package(s) the system may find. To achieve this, we adopt as system subgoals the set of tourist's preferences that the touristic packages are expected to satisfy. On another branch of the goal hierarchy, we consider the tourist's rejections. Since all the preferences (positive and negative) are at the same level in the hierarchy, we use different degrees to reflect the different importance the tourist gives to both, positive and negative desires. For example, we may have a user who is looking for a touristic package for his holidays. He has a strong desire of going to a mountain place (degree 0.8), to make rafting (degree 0.6) and rejecting to go to the northern Argentinian region (degree 0.9). The system then takes these desires as subgoals to find some packages satisfying these preferences and rejections.

4.3 System design: Defining agents

During an agent-based design, emphasis is put on defining software agents and on how they collaborate to fulfill the requirements.

In specifying an agent system, it has been found that is highly desirable to adopt specialized set of models which operate at two distinct levels of abstraction as in [7]. In one level, from the external viewpoint, the system is decomposed into agents, modeled by their roles, responsibilities and services they perform, the information they require and maintain, and their external interactions. In a second level, from the internal point of view, the elements required by a particular agent architecture must be modeled for each agent. In our case, these will be done using our graded BDI model of agency. Then, for its specification we need to set the different contexts for the agent's beliefs, desires and intentions, and the interactions between them, represented by bridge rules.

Step 4: Defining agents by role composition

Different agent-based software engineering methodologies take advantage of a role analysis for the agents definition as for example in [6, 11]. From the list of goals and plans, a responsibility and role analysis is held in order to define the set of agents and interactions that compose the system. Roles, responsibilities and goals (or services) are just descriptions of purposeful behaviors at different levels of abstraction: roles can be seen as a set of responsibilities and responsibilities as sets of goals. This process of role composition and the role assignment to agents is a difficult task because is not only a distribution problem. There is a need to compose roles because, when agents carry out their responsibilities and interact/collaborate, there may be a synergy between the different roles played by an agent. [11]

In our research, a role includes a set of goals and a set of plans towards them. Such a role will be mapped to an agent who is responsible for satisfying its corresponding goals. The activity of identifying roles from use cases and the use of role patterns in agent software engineering can be seen in [6]. These roles are candidate agents.

In agent-based modeling, Dynamic Diagrams imported from the object oriented modeling may be useful, for example, some sequence diagrams representing related actions for the plans in each use case, or activity diagrams expressing operations and the events that trigger agents [2].

Case Study:

For our case study we have detected two important roles: the tourist packages provider (Provider) and the finder of the most suitable tourist packages acording to the tourist prefeences (Travel Assistant). The Provider role has the responsabilities of building different packages, updating the list of touristic plans (charge-discharge) and sending messages to the Travel Assistant role with the package updates. Different Tourist Operators will collaborate in this task.

The Travel Assistant role is described in more detail using a Responsability, Goals and Collaborator (RGC) card as is shown in Table 1.

From this process of role identification and composition, the list of the candidate agents, which might then be refined or modified during the design process, can be obtained.

For this simplified version of our Recommender System, the conceptual agents are: The Travel

Responsabilities	**Goals**	**Collaborators**
Update Touristic Packages Reservory	Receive the message sent by Operators Update Packages Reservory	Operators
Find Feasible Packages	Identify the subsets of the set of preferences For each subset, find the packages satisfying it Take the union to be the set of Feasible Packages	Tourist Operators
Select best Feasible Packages	Estimate the benefit of each package Estimate the normalized cost of each package Estimate the benefit/cost relation Choose the best packages	Tourist

Table 1: RGC card for Travel Assistant role

Assistant Agent (T-Agent) and the Tourist Operator Agents (T-Operators) that supply the T-Agent with touristic packages.

Step 5: Internal Use Cases

The Internal Use Case concerns interactions among elements inside the system. and how they use each other to get things done. We propose a set of interacting agents which are assigned a particular role (or set of roles), where in turn a role specifies a particular goal (or set of goals). The purpose of this step is to identify the plans for each goal in the agent's role. Based on the external use cases, the scenario of each use case may be further detailed. This step will help us to find suitable agents plans to reach the positive desires avoiding the negative ones. Some authors as [5, 11] directly relate these plans to the notion of intentions in the agent design. We consider a more complex notion of the agent's intention that looks for the pair *feasible plan-desire* that best satisfies a cost/benefit relation. As a matter of example, we describe one internal use case for the *Look for Feasible Tourist Packages* functionality.

Case Study:

Name: Look for Feasible Tourist Packages
Actors: Tourist, T-Operators
Description:
- A Tourist asks the System for a recommendation of touristic plans in Argentinien destinations.
- The Systems gets the different graded preferences from the user (the system lists the possibilities the tourist may choose, the tourist also include degrees for them).
- The Systems gets the different graded rejections from the user (the system lists the possibilities the tourist may choose)
- The different T-Operators send the T-Agent the touristic packages they offer provided by them in a format established by the T-Agent.
- The system looks for feasible plans, each one satisfies a set of preferences (taking all the subsets from the set of preferences) and avoids all the rejections.
- A set of feasible packages passes to the *Select the best feasible packages* responsability.

The development of the internal use cases helps to better understand the interactions and collaborations between responsabilities and roles, and consequently, between the candidate agents. Then, the definition of agents by role assignment can be improved.

4.4 A Multicontext BDI Agent Design

Once we have defined the different agents in the system by means of the previous steps, we have to deal with their internal design. Namely, we must decide what kind of agent architecture is appropriate in each case according to its characteristics and the role assignment. The BDI paradigm provides a strong notion of agency: agents are viewed as having certain mental attitudes (Beliefs, Desires and Intentions) which represent respectively their information, motivational and deliberation states. These mental attitudes play a relevant role in the process of determining the agent's actions. The BDI model has proved to be a suitable choice to model complex agents, situated in dynamic and uncertain environments.

In the stage of an agent's internal design, we focus on describing the process of modeling graded BDI agents, which integrate a multiagent system. Taking advantage of the multicontext approach, this amounts to specifying the different contexts (either mental –belief, desire and intention, or functional –planner, communication) and the bridge rules. For the different contexts' modelization we will use the information acquired in the analysis and external design stages. The context design is done in two phases. First, we must take into account what the contents of each mental unit will be, and then, which is the appropriate logical formalism to represent this information and the reasoning process which is involved. In the following subsections we depict this internal design process, using a multicontext specification, for the Travel Assistant Agent in the Recommender System.

Desire Context (DC):

The different agents in the system were defined from role assignment, having a set of goals and plans (Step 4). These goals constitute the agent's basic positive desires. We also consider important for the agent to include a set of negative desires, representing its rejection states. As it was mentioned in the system analysis (Step 3), the set of desires (positive and negative) may have different levels of importance, represented as degrees in $[0, 1]$. We propose to use modal many-valued formulae to represent positive and negative desires (see [3] for details). Two (fuzzy) modal operators D^+ and D^- are introduced. $D^+\varphi$ reads as "φ is positively desired" and its truth degree represents the agent's level of satisfaction would φ become true. $D^-\varphi$ reads as "φ is negatively desired" and its truth degree represents the agent's measure of disgust on φ becoming true.

In our Case Study, in the DC the tourist's desires will be expressed by a theory T containing quantitative expressions about positive and negative preferences. These formulae express what the tourist desires (e.g. $(D^+(mountain), 0.8)$ or rejects (e.g. $(D^-(northregion), 0.9)$) in different degrees, for his holidays. These desires are the proactive elements of the recommender T-Agent and they start a chain of intra and inter-context deductions in order to determine which is the best touristic plan to recommend to the user.

Planner Context (PC):

Using Step 5, we find the set of actions that the agent may follow to reach a positive desire or set of desires, avoiding the negative ones. These plans are composed by the elementary actions that the agent can perform. The Planner context, a functional context, will be in charge of finding these feasible plans. We propose to use a first order language restricted to Horn clauses (PL), where a theory of planning includes at least special predicates for the actions and plans.

For our Travel Assistant Agent, the Planner context will be in charge of finding the feasible tourist packages that are expected to satisfy the tourist's preferences.

Belief Context (BC):

Beliefs represent the (uncertain) knowledge about the agent state and the changing environment. This knowledge is used to derive conclusions about whether plans may fulfill the agent's goals (desires). In order to specify which are the agent beliefs, we may apply some artifacts as the Data Flow Diagram, which show the flow of data as well as its logical storage. The agent's beliefs will be the needed information in relation to its goals assignment. For this design process the input and output data requirements for each subgoal in a plan must be analyzed. Depending on the agent's environment and how the agent gets the information from it, the agent knowledge about the world may be of different kinds: uncertain, imprecise, incomplete, etc. In the case of having uncertain information, we propose to use fuzzy modal formulae, following the logical framework described in Section 2 and to consider probability theory as the uncertainty model (see [3] for details).

In our Case of Study, the theory for the BC of the T-Agent contains general knowledge about the touristic domain, as the different Argentinian regions and destinations, the geographic characteristic of each region, activities allowed in each place, among others. We structure this knowledge inspired by some tourism ontologies. This context contains also the information about the touristic plans that the different operators provide. Fusthermore, this theory includes the relationship between the packages the T-Agent has in its repository and the preferences that are satisfied by their execution. The T-Agent has to represent the necessary knowledge to infer the beliefs about how the possible desires D (e.g. going to a mountain place or making rafting) may be satisfied after executing different plans α. Following the model presented, we consider the truth-degree of $B([\alpha]D)$ as the probability of D becoming true after following the plan α. We use classical formulae to represent part of the touristic information as for example, the distance between two destinations. Also, many-valued modal formulae are used to represent uncertain knowledge.

Intention Context (IC)

To complete the agent's design we need to specify how the agent's intentions will be determined. In our model the intention of an agent is a pair desire/plan (the desire it decides to follow through a plan) that is determined following a deliberation process. We consider that this attitude depends on different factors as the degree of the desires involved, the expected satisfaction degree of the desires through the plan execution and the cost of executing the plan. Different kinds of agents may be defined according to the way these different elements are weighted. This is formalized using a bridge rule. In the process of defining the agent intentions, the beliefs and desires of the agent are involved. So, we may define in parallel the necessary bridge rules that which enable to export formulae from one context to another.

For the IC we propose to represent two kinds of graded intentions, intention of a formula φ considering the execution of a particular plan α, noted $I_\alpha\varphi$, and the final intention to achieve φ, noted $I\varphi$, which takes into account the *best* path to reach φ. Then, for each plan α we introduce a modal operator I_α, and a modal operator I, in the same way as we did in the other contexts. The intention to make φ true must be the consequence of finding a *feasible* plan α, that permits to achieve a state of the world where φ holds. A theory for IC in the T-Agent represents those desires the user can intend by different feasible plans. Using this set of graded intentions, the T-Agent derives the final intention and the most recommended touristic plan. This theory is initially empty and will receive from a suitable bridge rule (see (1) in subsection Bridge Rules) formulae like $(I_\alpha\varphi, i)$ for all the desires φ and for all the feasible package α that the Planner finds.

Communication Context (CC)

This context makes it possible to encapsulate the agent's internal structure by having a unique and well-defined interface with the environment. As in the PC we propose to use classical first order logic. The theory inside this context will take care of the sending and receiving of messages to and from other agents in the Multi Agent society where our graded BDI agents live.

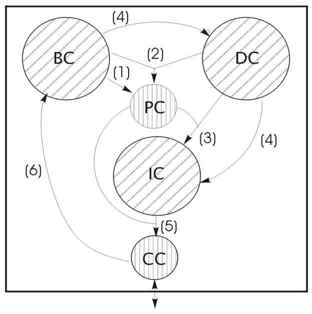

Figure 2: Multicontext model of a graded BDI agent

Bridge Rules

Bridge rules (BRs) represent the interaction between the different attitudes. As they put into relation different contexts, using different logics, these rules establish how a formula of a context can be embedded or can be used to derive a new formula in another unit. The design of the bridge rules is done simultaneously with the process to determine the agent's intentions. The necessary BRs for an agent's specification will depend on its role assignment. As for example, the set of positive and negative desires must be passed to the Planner context which is in charge to find the feasible plans to satisfy these desires. Some beliefs are also needed by the PC context to find these plans, like the elementary actions that the agent can execute, or the expected satisfaction of the different desires after a plan implementation.

For the T-Agent we have defined a set of bridge rules, analizing the inter-context inferences. As for example, there is a bridge rule that infers the degree of $I_\alpha\varphi$ for each feasible plan α that allows to achieve the goal φ. This value is deduced from the degree of $D^+\varphi$, the degree r of belief $B[\alpha]\varphi$ and the cost of the plan α. These graded intentions are deduced by the following rule using an adequate function f:

$$\frac{DC : (D^+\varphi, d), PC : fplan(\varphi, \alpha, P, A, r, c)}{IC : (I_\alpha\varphi, f(d, r, c))} \tag{1}$$

Step 6: Specifying graded BDI Agents

Collecting the specification of the different contexts (BC, DC, IC, PC and CC) and the bridge rules (BRs) , the multicontext specification for the graded BDI agent is complete. Figure 2 illustrates this agent architecture.

5 CONCLUSIONS AND FUTURE WORK

Agent-based computing has increased in the last years and thus developing software engineering methodologies to build these systems has become an urgent need. Even though there are valuable approaches in this field, few of them emphasize the internal design of agents, considering different architectures. In this paper we have presented some contributions in this direction, proposing a software engineering process to develop graded BDI agents in a multiagent scenario. The methodology presented has been built adapting and extending previous approaches [5, 11] in order to engineer agents with a more complex internal architecture. Our methodology extracts the necessary elements for the design of BDI agents following a flow “goals-feasible plans-beliefs-intentions” and presenting a new process to obtain the agents' intentions. In this sense, we have modified the flow “goals-plans(intentions)-beliefs” used in the mentioned approaches. Particularly, we have presented the methodology applied to the design of graded BDI agents, extending the BDI model with the capabitities of dealing with the environment uncertainty and with graded mental attitudes, using MCS for agent's specification. One advantage of this logical approach to agency is that allows for a rather affordable computational implementation.

Completing the development of the case study, we are now implementing a prototype of the Travel Assistant Agent using a multi-threaded version of Prolog. This development process, from the system requirements to implementation, will help us to revise all the stages of the methodology presented. As for future work, we plan to use this methodology in other case studies in order to improve it.

Acknowledgments: The authors are indebted to Juan Antonio Rodríguez for his helpful comments. They also acknowledge for partial support of the projects PCI-IBEROAMERICA (A/3541/05), WEBI2 (TIC2003-08763-C02-00) and MULOG (TIN2004-07933-C03-01).

REFERENCES

[1] Bergenti F., Gleizes M.P. and Zambonelli F., Eds., *Methodologies and Software Engineering For Agent Systems: The Handbook of Agent-oriented Software Engineering*, Kluwer Academic Publishing (New York, NY), July 2004.

[2] Booch G., Rumbaugh J. and Jacobson I. The Unified Modelling Language. Addison-Wesley, 1999.

[3] Casali A., Godo L. and Sierra C. Graded BDI Models For Agent Architectures. J. Leite and P. Torroni (Eds.) *CLIMA V, LNAI 3487*, 126-143, 2005.

[4] Casali A., Godo L. and Sierra C. Multi-Context Specification for Graded BDI Agents. Proceedings of *CONTEXT-05*, Research Report LIP 6, Paris, 2005.

[5] Jo Ch.H., Chen G. and Choi J. A New Approach to the BDI Agent-BAsed Modeling. *ACM Symposium on Applied Computing SAC'04*, ACM 1-58113-812-1/03/04, 1541-1545, 2004.

[6] Kendall E. A. Agent Software Engineering with Role Modeling. In Proceedings of *Agent Oriented Software Engineering AOSE-2000*Ciancarini P. and Wooldridge M. (eds.), Springer-Verlag, Berlin, Germany, 2000.

[7] Kinny D., Georgeff M., and Rao A. A Methodology and Modelling Techniques for Systems of BDI agents. Proc. of the 7*th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, (LNAI Vol. 1038)*: 56-71, Springer, 1996.

[8] Rao A. and Georgeff M. Modeling Rational Agents within a BDI-Architecture. In *KR-92*, 473-484 (ed R. Fikes and E. Sandewall), Morgan Kaufmann, 1991.

[9] Sabater J., Sierra C., Parsons S. and Jennings N. R. Engineering executable agents using multi-context systems. *Journal of Logic and Computation*12(3): 413-442, 2002.

[10] Wooldridge M. J., Jennings N.R. and Kinny D. A Methodology for Agent-oriented Analysis and Design, Proceedings of the *Third International Conference on Autonomous Agents (Agents'99)* 3(3), pp285-312. September 2000.

[11] Zhang T., Kendall E. and Jiang H. A Software Engineering Process for BDI Agent-Based Systems, in Proceedings of the *IEEE/WIC International Conference on Intelligent Agent Technology (IAT03)*. 0-7695-1931-8/03IEEE, 2003