Developing Intentional Systems with the PRACTIONIST Framework

Vito Morreale*, Susanna Bonura*, Giuseppe Francaviglia*, Fabio Centineo*,
Michele Puccio*, and Massimo Cossentino†§
*R&D Laboratory - ENGINEERING Ingegneria Informatica S.p.A.
†ICAR-Italian National Research Council
§SET - Universit de Technologie Belfort-Montbliard, France

***Abstract*— Agent-based systems have become a very attractive approach for dealing with the complexity of modern software applications and have proved to be useful and successful in some industrial domains. However, engineering such systems is still a challenge due to the lack of effective tools and actual implementations of very interesting and fascinating theories and models. In this area the so-called intentional stance of systems can be very helpful to efficiently predict, explain, and define the behaviour of complex systems, without having to understand how they actually work, but explaining them in terms of some mental qualities or attitudes, rather than in terms of their physical or design stance.**

In this paper we present the PRACTIONIST framework, that supports the development of PRACTIcal reasONIng sySTems according to the BDI model of agency, which uses some mental attitudes such as beliefs, desires, and intentions to describe and specify the behaviour of system components. We adopt a goal-oriented approach and a clear separation between the deliberation and the means-ends reasoning, and consequently between the states of affairs to pursue and the way to do it. Moreover, PRACTIONIST allows developers to implement agents able to reason about their beliefs and the other agents' beliefs, expressed by modal logic formulas.

I. Introduction

Several authors (e.g. McCharty [1]) have argued that in certain situations, the so-called intentional stance [2] of systems can aid to efficiently predict, explain, or define their behaviour, without having to understand how they actually work. Therefore, some systems may be better explained in terms of mental qualities or attitudes, rather than in terms of conventional physical phenomena or design artifacts.

In the context of the development of intentional systems, the agent-oriented approach plays a central role, due to the vast number of theories and models that have been developed for twenty years. One of the most interesting and attractive agent models is the Belief-Desire-Intention (BDI) architecture [3], which suggests that the development of agents should rely on the specification of some mental states, i.e. beliefs, desires, and intentions, which are very intuitive for people to understand. Indeed, beliefs represent information the agent has about the world; desires represent state of affairs the agent wishes to bring about and intentions are desires that it has committed to achieving.

The BDI model of agency derives from the philosophical tradition of practical reasoning, which states that any agent decides, moment by moment, which actions to perform in order to pursue his/her goals. Practical reasoning involves two processes: (1) *deliberation*, to decide *what* states of affairs to achieve; and (2) *means-ends reasoning*, to decide *how* to achieve these states of affairs. In such a theory intentions are important, as they influence the selection of the actions to perform.

In spite of the well-established and fascinating theory underlying the BDI model, the development of this agent architecture involves several issues regarding the efficient implementation of the deliberation process and the means-ends reasoning [4].

The most-known BDI agent abstract architecture is the Procedural Reasoning System (PRS), developed by Georgeff and Lansky [5]. Some concrete implementations of PRS have been developed, such as dMARS [6], developed at the Australian AI Institute, the UM-PRS implemented in C++ at the University of Michigan [7], and a Java version of PRS called JAM [8]. Finally, it is worth mentioning JACK [9], which is a commercially available programming language that extends the Java language with BDI features.

However in several BDI implementations, mental states, deliberation, and means-ends reasoning somewhat differ from their original meaning. As an example, often executing plans are considered as intentions. But we believe that intentions should be related to ends, while plans should be related to means to achieve such ends.

In this paper we present the PRACTIONIST framework, that supports the development of PRACTIcal reasONIng sySTems according to the BDI model of agency. We wanted to stress the separation between the deliberation process and the means-ends reasoning in PRACTIONIST agents. Thus, plans are recipes to achieve the intentions, rather than intentions themselves. Here the abstraction of goal is used to formally define both desires and intentions during the deliberation phase. The explicit representation of agents' potential objectives (i.e. goals) and the relations among them provides PRACTIONIST agents with the ability to figure out if such objectives are impossible, achieved, incompatible with one another, and so forth [10]. This supports commitment strategies to agents' intentions. Moreover, PRACTIONIST agents are able to reason about their beliefs and the other agents' (including humans') beliefs.

It should be noted that some useful intentional agent patterns can directly and actually be implemented with the PRAC-

TIONIST framework (refer to [11] for details). In other words, in our framework we wanted to solve some of most common design problems at the agent level, by providing some built-in solutions that programmers can easily adopt when developing their agents.

Our definitive intention is to define a complete set of tools to efficiently and effectively develop complex and adaptive distributed software applications based on a more flexible and robust architecture of intentional components than the one underlying other BDI-based platforms.

Thus, the PRACTIONIST framework is part of a suite of tools we have been developing, which also includes a methodology (consisting of a UML-based modelling language, a development process, some techniques and guidelines) to design and implement BDI agent-based systems according to the PRACTIONIST model and a design and development tool supporting such a methodology. The methodology and the tool are out of the scope of this paper, that mainly focuses on the PRACTIONIST framework.

This paper is organized as follows: in section II a general overview of the framework is given; section III describes the model of beliefs in PRACTIONIST agents; then the deliberation process (section IV) and the means-ends reasoning (section V) are presented; and finally the development of two real-world applications is briefly discussed (section VI).

II. PRACTIONIST FRAMEWORK

The PRACTIONIST framework supplies the support for developing BDI agents *(i)* endowed with a symbolic representation about their beliefs, *(ii)* able to proactively deliberate about their intentions, (iii) capable of performing reactive behaviours, and *(iv)* provided with the ability to plan their activities in order to pursue some objectives.

PRACTIONIST has been designed on top of JADE, a widespread platform compliant to the FIPA[1] specifications, that provides some core services, such as a communication support, interaction protocols, life-cycle management, and so forth. Therefore, the PRACTIONIST agents are executed within JADE containers and the main cycle is implemented by means of a JADE cyclic behaviour.

As shown in figure 1, PRACTIONIST agents are structured in two main layers: the framework defines the execution logic and provides some built-in services implementing such a logic, while the top layer includes the specific agent components to be defined in order to satisfy specific application requirements.

Therefore, a developer who wants to design a PRACTIONIST agent has to specify *(i)* the *Goals* the agent could pursue and the relations among them (i.e. the Goal Model), *(ii)* the means (a set of plans, i.e. the *Plan Library*) to pursue such goals or to react to the stimuli coming from the environment, *(iii)* the *Perceptors* to receive such stimuli, *(iv)* the *Actions* the agent could perform and the corresponding *Effectors*, and *(v)* the set of beliefs and rules (*Belief Base*) to model the information about both its internal state and the external world.

Then the framework provides the required built-in services that define the computational model of PRACTIONIST agents.

[1]http://www.fipa.org

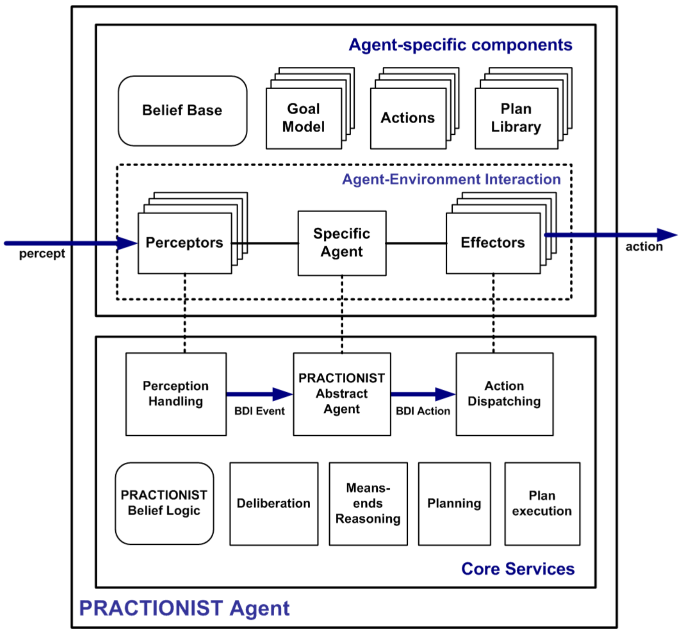


Fig. 1. PRACTIONIST Agent Architecture.

This includes the *Belief Logic* (as described in section III), the *Deliberation* mechanisms that produce agent intentions, the way the agent makes *Means-ends Reasoning* to figure out the means (i.e. plans) to achieve its intentions, and the support for the actual execution of such plans.

Moreover, agents are endowed with to dynamically build plans (i.e. *Planning*). Finally the management of perceptors and effectors is part of the agent *core services* infrastructure.

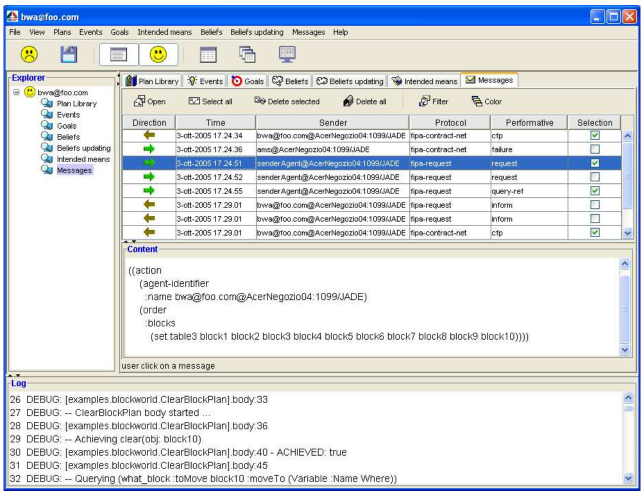
The framework also comes with the PRACTIONIST Agent Introspection Tool (PAIT), a visual integrated monitoring and debugging tool, which supports the analysis of the agent's state during its execution. In particular, the PAIT can be suitable to display, test and debug the agents' mental attitudes (i.e. beliefs, desires, and intentions) and their execution flow, in terms of active behaviours. Each of these components can be observed at run-time through a set of specific tabs (see figure 2).

All the collected and displayed information could be saved in a file, providing programmers with the possibility to perform an off-line analysis. Moreover, the PAIT provides a dedicated area for log messages inserted in the agent source code, according to the Log4j[2] approach. It should be also noted that this feature of PAIT is very useful when developing and testing agents, as it provides the user with a real-time snapshot of agents' attitudes.

III. BELIEFS

Since agents' information about the world is usually incomplete or incorrect, due to uncertainty and problems with perceptions and communication in dynamic and possibly unpredictable environments, the BDI model refers to *beliefs* instead of knowledge. Indeed beliefs are not necessarily true, while *knowledge* usually refers to something that is true [12]. According to this, an agent may believe true something that is

[2]http://logging.apache.org/log4j



bwa@foo.com
File View Plans Events Goals Intended means Beliefs Beliefs updating Messages Help
Explorer
bwa@foo.com
Plan Library
Events
Goals
Beliefs
Beliefs updating
Intended means
Messages
Plan Library Events Goals Beliefs Beliefs updating Intended means Messages
Open Select all Delete selected Delete all Filter Color
Content
Log

Direction	Time	Sender	Protocol	Performative	Selection
←	3-ott-2005 17.24.34	bwa@foo.com@AcerNegozio04:1099/JADE	fipa-contract-net	cfp	✓
→	3-ott-2005 17.24.36	ams@AcerNegozio04:1099/JADE	fipa-contract-net	failure	☐
→	3-ott-2005 17.24.51	senderAgent@AcerNegozio04:1099/JADE	fipa-request	request	✓
→	3-ott-2005 17.24.52	senderAgent@AcerNegozio04:1099/JADE	fipa-request	request	☐
→	3-ott-2005 17.24.55	senderAgent@AcerNegozio04:1099/JADE	fipa-request	query-ref	✓
←	3-ott-2005 17.29.01	bwa@foo.com@AcerNegozio04:1099/JADE	fipa-request	inform	☐
←	3-ott-2005 17.29.01	bwa@foo.com@AcerNegozio04:1099/JADE	fipa-request	inform	☐
←	3-ott-2005 17.29.01	bwa@foo.com@AcerNegozio04:1099/JADE	fipa-contract-net	cfp	✓

```
((action
   (agent-identifier
     :name bwa@foo.com@AcerNegozio04:1099/JADE)
   (order
     :blocks
       (set table3 block1 block2 block3 block4 block5 block6 block7 block8 block9 block10))))
```

user click on a message

```
26  DEBUG: [examples.blockworld.ClearBlockPlan].body:33
27  DEBUG: -- ClearBlockPlan body started ...
28  DEBUG: [examples.blockworld.ClearBlockPlan].body:36
29  DEBUG: -- Achieving clear(obj: block10)
30  DEBUG: [examples.blockworld.ClearBlockPlan].body:40 - ACHIEVED: true
31  DEBUG: [examples.blockworld.ClearBlockPlan].body:45
32  DEBUG: -- Querying (what_block :toMove block10 :moveTo (Variable :Name Where))
```

Fig. 2. The PRACTIONIST Agent Introspection Tool (PAIT).

false from the other agents' (including humans') point of view. In other words we want to provide agents with a subjective window upon the world.

In PRACTIONIST, beliefs can be about either propositions or other beliefs. In other words, an agent may believe or not something described as a proposition (e.g. "it is snowing in Paris"). It may also believe that some agent (it could be itself) believes something (e.g. "the agent John believes that it is snowing in Paris").

In the PRACTIONIST framework beliefs are expressed through the modal operator

$$Bel(\alpha,\ \varphi)$$

which has two arguments, i.e. the agent α (the believer) and what it believes (φ, the fact). More formally, let F be a set of closed formulas of first-order logic (FOL) and $\hat{F}$ the set of formulas of the modal logic[3] based on F. Then, the fact believed by an agent is expressed by elements of $\hat{F}$, such as φ, $\Diamond\vartheta$, $\Box\Diamond\tau$, $\forall x\, \psi(x)$ etc., where φ, ϑ, and τ are FOL closed formulas belonging to F, ψ is a predicate symbol, and $\Box$ and $\Diamond$ are respectively the classical modal operators of *necessitation* and *possibility*.

In PRACTIONIST, predicates can also be represented by specifying the role of their arguments, as follows:

$$predicate(role1 : element1, ..., roleN : elementN).$$

Moreover, we defined a special unification function based on the role of terms within a predicate, instead of their order. As an example, $fatherOf(father : jim, son : bob)$ and $fatherOf(son : bob, father : jim)$ represent the same fact.

We also assume the KD45 axioms corresponding to a "Weak S5 modal logic" (see [12] for more details), which for any agent α asserts that:

- (K) $Bel(\alpha, \varphi \Rightarrow \psi) \Rightarrow (Bel(\alpha, \varphi) \Rightarrow Bel(\alpha, \psi))$
- (D) $Bel(\alpha, \varphi) \Rightarrow \neg Bel(\alpha, \neg\varphi)$
- (4) $Bel(\alpha, \varphi) \Rightarrow Bel(\alpha, Bel(\alpha, \varphi))$
- (5) $\neg Bel(\alpha, \varphi) \Rightarrow Bel(\alpha, \neg Bel(\alpha, \varphi))$

For any $\varphi \in \hat{F}$, a PRACTIONIST agent α may express or infer three distinct belief states:

- $Bel(\alpha, \varphi)$: the agent α believes that φ is true;
- $Bel(\alpha, \neg\varphi)$: the agent α believes that φ is false;
- $Ubif(\alpha, \varphi) \equiv \neg Bel(\alpha, \varphi) \wedge \neg Bel(\alpha, \neg\varphi)$: the agent α does not have any belief about φ.

In PRACTIONIST it is also possible to define *nested beliefs*, such as $Bel(\alpha, Bel(\beta, \varphi))$ and $Ubif(\alpha, Bel(\beta, \neg\varphi))$, which respectively mean that the agent α believes that the agent β believes that φ is *true* and the agent α does not have any belief about the fact that the agent β believes that φ is *false*.

Each PRACTIONIST agent is endowed with a Prolog belief base, where beliefs are asserted, removed, or entailed through inference on the basis of KD45 rules and user-defined belief rules. Therefore, in any moment the agent's belief set is composed of the beliefs that have been both directly asserted and inferred by means of rules and the other built-in theorems. Currently the PRACTIONIST framework supports two prolog engines, i.e. SWI-Prolog[4] and a customization of TuProlog[5].

IV. Deliberation Process

In this section we describe the structure of PRACTIONIST goal model and how an agent uses it during the deliberation phase, in order to select intentions to achieve.

The PRACTIONIST framework adopts a goal-oriented approach to develop BDI agents and stresses the separation between the deliberation process and the means-ends reasoning, with the abstraction of goal used to formally define both desires and intentions during the deliberation phase. Therefore PRACTIONIST agents can be programmed in terms of goals, which then will be related to either desires or intentions according to whether some specific conditions are satisfied or not.

Unlike some available BDI agent platforms, the PRACTIONIST framework supports the explicit representation and implementation of goals with their properties and relations. This provides the ability to reason about them, in terms of *believing* if goals are impossible, achieved, incompatible with other goals, and so forth. This in turn supports the *commitment strategies* of agents and their capability to autonomously drop, reconsider, replace or pursue goals.

Formally, a *goal* g is defined as follows:

$$g = \langle \sigma_g,\ \pi_g,\ \gamma_g \rangle \tag{1}$$

where σ_g is the *success condition*, π_g is the *possibility condition* stating whether g can be achieved or not, and γ_g is the *cancel condition* stating in which situations the agent should give up to pursue the goal g.

With the aim of defining the goal model of PRACTIONIST agents, we first provide the following definitions regarding some useful relations among goals:

- a goal g_1 is *inconsistent* with a goal g_2 if and only if when g_1 succeeds, then g_2 fails;

[3]The syntax of the modal logic is defined by the following rules: (1) if $\varphi \in F$ then φ is a formula (that is $\varphi \in \hat{F}$); (2) if φ is a formula then so are $\Box\varphi$ (necessarily φ) and $\Diamond\varphi$ (possibly φ).

[4]http://www.swi-prolog.org

[5]http://tuprolog.alice.unibo.it

- a goal g_1 *entails* a goal g_2 (or equivalently g_2 is *entailed by* g_1) if and only if when g_1 succeeds, then also g_2 succeeds;
- a goal g_1 is a *precondition* of a goal g_2 if and only if, to be possible to pursue g_2, g_1 must succeed;
- a goal g_1 *depends* on a goal g_2 if g_2 is precondition of g_1 and g_2 must be successful while pursuing g_1.

When two goals are inconsistent with each other, we can also specify that one is preferred to the other. In PRACTIONIST several goals can be pursued in parallel. Therefore there is no need to prefer some goal to another goal if they are not inconsistent with each other.

The *goal model* of PRACTIONIST agents contains the set of goals they could pursue and all above-defined relations among such goals (i.e. inconsistence, entailment, precondition, and dependence). A more formal definition of the goal model can be found in [13].

The goal model is used by PRACTIONIST agents when reasoning about goals during their deliberation process. For them, desires and intentions are mental attitudes towards goals, which are in turn considered as descriptions of objectives. Thus, ”pursuing the goal g” is only a desire if the agent is not yet committed to it, due to some reason. On the other hand, ”pursuing the goal g” becomes an intention when the agent is committed to it and works to achieve it.

Suppose that an agent starts its deliberation process (see figure 3) and generates a goal $g = \langle \sigma_g,\ \pi_g,\ \gamma_g \rangle$ as an option. Therefore the agent *desire*s to pursue the goal g. But any agent will not be able to achieve all its desires; thus it checks if it believes that the goal g is *possible* (i.e. if it believes that π_g is true) and not *inconsistent* with active goals (i.e. those goals that the agent is currently committed to), the desire to pursue g will be promoted to an *intention*. Otherwise, in case of inconsistence among g and some active goals, the desire to pursue g will become an intention only if g is preferred to all inconsistent active goals, which will in turn be dropped.

Therefore, at the end of the deliberation process, a PRACTIONIST agent could either generate a new intention or remain with an impossible desire; it could drop some existing intentions as well. This ability avoids that agents try to pursue impossible and inconsistent goals (at least from their point of view).

When a desire to pursue g is promoted to an *intention*, before starting the means-ends reasoning, the agent checks if it believes that the goal *g succeeds* (that is, if it believes that the success condition σ_g holds) or if the goal *g is entailed* by some of the current active goals (i.e. some other means is working to achieve a goal that entails the goal g). If either, there is no reason to pursue the goal g and the agent does not need to make any means-ends reasoning to figure out how to pursue it.

Otherwise, before starting the means-ends reasoning, if some declared goals are precondition for g, the agent will first desire to pursue them and then the goal g.

As a default, PRACTIONIST agents adopt a *single-minded intention commitment* strategy. Thus, it will continue to maintain an intention until it believes that either such an intention has been achieved or it is no longer possible to achieve the intention. Moreover the agent checks if some dependee goal does not succeed. If so, it will desire to pursue it and then continue pursuing the goal g.

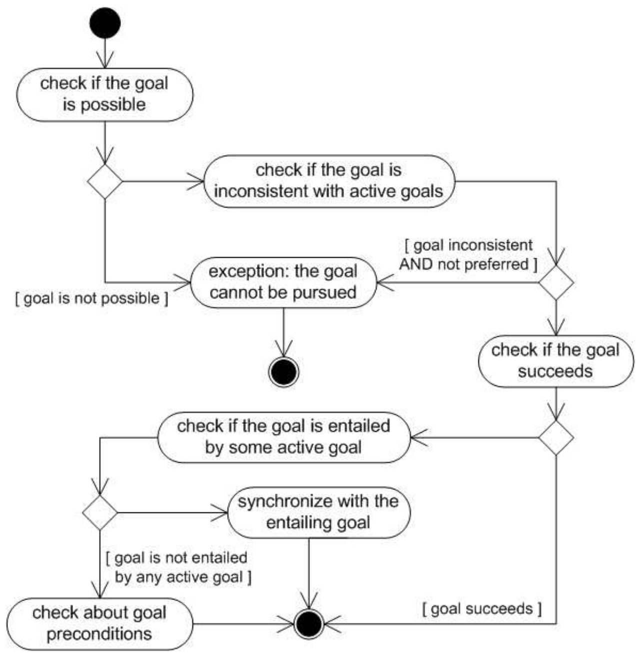

check if the goal is possible
check if the goal is inconsistent with active goals
[goal inconsistent AND not preferred]
exception: the goal cannot be pursued
[goal is not possible]
check if the goal succeeds
check if the goal is entailed by some active goal
synchronize with the entailing goal
[goal is not entailed by any active goal]
check about goal preconditions
[goal succeeds]

Fig. 3. The deliberation process.

V. Means-Ends reasoning

In the PRACTIONIST framework plans represent an important container where it is possible to define the actual behaviours of agents. Each agent may own a declared set of plans (the *plan library*), which specify the activities the agent should undertake in order to achieve its intentions, or handle incoming perceptions, or react to changes of its beliefs.

Several information about plans can be specified (the complete list of such slots is reported in table I), in order to provide the agents with the capability to dynamically behave when selecting and executing plans. Thus, a plan represents a possible recipe to manage the trigger event, which in turn may be related to a goal, an external event, or an event which notifies a change of the belief set. How to actually handle a certain event is reported within the body, which is an *activity* that can contain a set of *acts*, such as *desiring* to pursue some goal, *adding* or *removing* beliefs, *sending* ACL messages, *doing* an action and so forth.

Through the perceptors, a PRACTIONIST agent receives perceptions from the environment and transforms them into *events*, which are put into an event queue. It also contains internal goal events, which are generated when the desire to pursue a given goal is promoted to an intention and some means-ends reasoning is required to figure out how to achieve such an intention. The queue also collects events related to changes in agent’s beliefs.

For each event e extracted from the queue, the agent performs the following *means-ends reasoning*:

1) it figures out the *practical* plans, which are those plans whose trigger event matches the selected event;

TABLE I
THE STRUCTURE OF PRACTIONIST PLANS.

Trigger event	The event (or the set of events) the plan is supposed to handle. It represents the *intent* of the plan.
Context	A modal logic formula that, when believed true by the agent, makes the plan *applicable*, so that the agent can select it.
Success condition	When the agent believes that this condition holds, the plan ends with success, regardless its execution state.
Cancel condition	When the agent believes that this condition holds, the plan ends with failure, regardless its execution state.
Body	Set of acts that are performed during the execution of the plan. The body defines the actual behaviour of the plan.
Invariant	Condition that must remain true during the execution of the plan. As soon as it becomes false (at least according to the agent's point of view), the agent will try to restore it.
Belief updates in case of success	Effects of this plan, in terms of belief updates in case of plan success.
Belief updates in case of failure	Effects of this plan, in terms of belief updates in case of plan failure.

2) among practical plans, the agent detects the *applicable* ones, which are those plan whose context is believed true, and selects one of them (which is called *main plan*);
3) it builds the *intended means*, containing the main plan and the other alternative practical plans.

Each new intented means is put within a stack according to the following criteria: if the event e refers to an intention to pursue a goal, it is put on top of the stack containing the intended means that has generated the commitment to that intention; otherwise, a new stack is created with the new intented means. Thus, every intended means stack can contain several nested intended means, each able to handle a given event, possibly through several alternative plans.

For each stack, the main plan of the topmost intended means is executed. Meanwhile, both success and cancel conditions of the plan are checked, in order to stop the execution (either with success or failure) before its normal completion.

Moreover all intended means stacks are *concurrently* executed, so that each PRACTIONIST agent can perform several non-inconsistent activities in parallel. Indeed, every PRACTIONIST agent will never perform activities or pursue goals incompatible with one another, at least according to its beliefs and goal model.

In order to be able to recover from *plan failures* and try other means to achieve an intention, if an executing main plan fails or is no longer appropriate to achieve the intention, then the agent selects one of applicable *alternative plans* within the same intended means and executes it.

If none of the alternative plans is able to successfully pursue the goal g, the agent takes into consideration the goals that *entail* g. Thus the agent selects one of them and considers it as a desire, processing it in the way described above, from deliberation to means-ends reasoning.

If there is no plan to pursue alternative goals, the achievement of the intention has failed, as the agent has not other ways to pursue its intention. Thus, according to agents beliefs, the goal was *possible*, but the agent was no able to pursue it (i.e. there are no plans).

It should be noted that the PRACTIONIST framework provides agents with the ability to dynamically build plans to pursue a given goal, as soon as no plan of the library is able to pursue it. These planning capabilities are based on a backward search algorithm in the states space [14] implemented in Prolog. Thus, the issue of pursuing a goal can be viewed as a planning problem where the initial state is represented by the agent's beliefs and the domain is represented by available actions. The planning component produces a sequence of actions to be performed in order to satisfy the goal. These actions will be part of the body of a dynamically-generated plan, which temporarily becomes an element of the agent's plan library. This plan may be composed by a set of either abstract or concrete actions, according to whether they have some inputs are specified or not. In abstract actions, inputs will be initialized with some outputs of previous actions.

As soon as each action is performed, its postconditions are applied to update the agent's beliefs. The actual effects of the overall plan should be the satisfaction of the initial goal (the ends), as the plan is the right means to pursue it.

The plan can fail when action preconditions are not satisfied due to some unexpected changes of beliefs or the execution of some action fails.

It should be noted that since the planning component could be time-consuming, the developer can disable it. Moreover, the developer can set the maximum number of actions for dynamically-generated plans.

VI. SOME REAL-WORLD APPLICATIONS

In this section we present some real-world software applications developed with the PRACTIONIST framework, i.e. the *eContract Negotiation System* for the automatic negotiation of electronic contracts between autonomous entities with a minimum human intervention, and the *Bulletin Board Monitoring System*, which monitors a sort of bulletin board on behalf of users in order to discover interesting offers/requests for goods.

The general architecture of the *eContract Negotiation System* is decomposed into three high-level components: *(i)* the *Buyer subsystem*, interfacing with parties playing the role of buyers; *(ii)* the *Seller subsystem*, interfacing with parties playing the role of sellers; and *(iii)* the *Negotiation Manager*, which monitors, supports, and controls the negotiations carried out within its scope.

The above-mentioned subsystems were designed as agents interacting among one another and using some artifacts within the negotiation environment. Negotiation agent types (i.e. *Buyer* and *Seller*) are endowed with some strategies that let them achieve some given objectives, according to their beliefs, representing user preferences and needs as well as negotiation state, opportunities, and business rules. Beliefs of the *Negotiation Manager* agent type mainly refer to the negotiation status.

Obviously goals are used to model agent objectives (e.g. achieving an agreement, formalizing a contract, etc.), while plans are used to implement actual behaviours of the agents (e.g. negotiation strategy, evaluation of proposals, choice of the best proposal, etc.).

It should be noted that all phases of eContract negotiation were addressed, such as the creation and the dispatch of proposals, the evaluation of received proposals, the dispatch of counterproposals and the formalization of agreements as formal eContracts.

Another application developed by using the PRACTIONIST framework is an automatic system that monitors offers and requests that have been published into a bulletin board and let users be informed about interesting ones, on the basis of their preferences and needs.

The general architecture of the *Bulletin Board Monitoring System* is decomposed into three high-level components: *(i)* the *Distributor subsystem*, interfacing with distribution companies (usually big enterprises) searching for interesting offers of goods; *(ii)* the *Producer subsystem*, interfacing with production companies (usually small and medium enterprises) searching for interesting requests of goods; and *(iii)* the *Bulletin Board*, which stores and provides collected requests and offers coming from producers and distributors.

The first two subsystems were designed as agents interacting among one another and using some artifacts, the most important of which is the bulletin board. Distributor and Producer agent types are endowed with some strategies that let them achieve the objectives given by corresponding owner (either a distributor or a producer), according to their beliefs, representing preferences and needs as well as opportunities and business rules.

It should be noted that in both above-mentioned applications the definition of the goal model of agents provides them with a high degree of flexibility and adaptability, as they can select the best strategy in several context conditions. Moreover the goal model provides the means to avoid the deliberation of impossible or incompatible intentions and to make agents more ”rational” (or equivalently less ”blind”) while pursuing their objectives.

VII. Conclusions and Future Work

In this paper we presented PRACTIONIST, a framework we have been developing to implement agent systems according to the BDI model, which provides developers with usable practical reasoning abstractions and processing facilities.

As described in previous sections, the PRACTIONIST framework supports the development of agents endowed with a lot of useful built-in capabilities, whereas several other platforms come with some of them. Thus, we strongly believe that the computational model of our agents is more flexible and adaptive than the agent models underlying several commercial and non-commercial frameworks. Moreover, PRACTIONIST agents are easy to design and implement, due to the vast set of built-in reasoning and effecting mechanisms they are endowed with.

Moreover, our framework provides a very expressive way to represent and reasoning about beliefs through modal logic formulas. The framework also allows agents to dynamically build plans in case of no plan available in the library can be activated by some selected event.

Some further work should be done with respect to the several issues that a BDI model involves; our intention is to improve the execution flow by adding some functionalities like timing, new acts, and so forth, that could help in the successful application of our framework in real problems.

Finally, we have been developing some other real-world applications by using the PRACTIONIST framework, methodology and design tool. As an example, we are designing and developing a software system for financial risk management as a society of PRACTIONIST agents, dealing with the problem of monitoring huge volumes of information and supporting humans in their decision-making in a distributed way.

VIII. Acknowledgment

This work is partially supported by the Italian Ministry of Education, University and Research (MIUR) through the project PASAF.

References

[1] J. McCarthy, “Ascribing mental qualities to machines,” Stanford University, Tech. Rep. STAN-CS-79-725, 1979.

[2] D. Dennett, *The Intetional Stance*. MIT Press, 1989.

[3] A. S. Rao and M. P. Georgeff, “Modeling rational agents within a BDI-architecture,” in *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, 1991, pp. 473–484. [Online]. Available: http://citeseer.nj.nec.com/rao91modeling.html

[4] G. Weiss, Ed., *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, 1999. [Online]. Available: http://jmvidal.cse.sc.edu/library/WeissBook/

[5] M. P. Georgeff and A. L. Lansky, “Reactive reasoning and planning,” in *Proc. of AAAI-87*, Seattle, WA, 1987, pp. 677–682.

[6] M. d’Inverno, D. Kinny, M. Luck, and M. Wooldridge, “A formal specification of dMARS,” ser. LNAI, M. P. Singh, A. Rao, and M. J. Wooldridge, Eds., vol. 1365. Springer, 1997, pp. 155–176.

[7] J. Lee, M. J. Huber, P. G. Kenny, and E. H. Durfee, “UM-PRS: An Implementation of the Procedural Reasoning System for Multirobot Applications,” in *Conference on Intelligent Robotics in Field, Factory, Service, and Space (CIRFFSS)*, Houston, Texas, 1994, pp. 842–849. [Online]. Available: citeseer.ist.psu.edu/lee94umprs.html

[8] M. J. Huber, “Jam: a bdi-theoretic mobile agent architecture,” in *AGENTS ’99: Proceedings of the third annual conference on Autonomous Agents*. New York, NY, USA: ACM Press, 1999, pp. 236–243.

[9] P. Busetta, R. Rnnquist, A. Hodgson, and A. Lucas, “JACK intelligent agents - components for intelligent agents in java,” *Agentlink News*, January 1999.

[10] M. Winikoff, L. Padgham, J. Harland, and J. Thangarajah, “Declarative & procedural goals in intelligent agent systems,” in *Proceedings of the Eighth International Conference on Principles of Knowledge Representation and Reasoning*, Toulouse, France, 2002, pp. 470–481.

[11] V. Morreale, G. Francaviglia, F. Centineo, M. Puccio, and M. Cossentino, “Goal-oriented agent patterns with the PRACTIONIST framework,” in *Proceedings of the Forth European Workshop on Multi-Agent Systems (EUMAS’06)*, Lisbon, Portugal, 2006.

[12] B. F. Chellas, *Modal Logic: An Introduction*. Cambridge: Cambridge University Press, 1980.

[13] V. Morreale, S. Bonura, G. Francaviglia, F. Centineo, M. Cossentino, and S. Gaglio, “Goal-oriented development of BDI agents: the PRACTIONIST approach,” in *Proceedings of Intelligent Agent Technology*. Hong Kong, China: IEEE Computer Society Press, 2006.

[14] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd ed. Prentice-Hall, Englewood Cliffs, NJ, 2003.