Reasoning about Goals in BDI Agents: the PRACTIONIST Framework

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

***Abstract*— The representation of goals and the ability to reason about them play an important role in goal-oriented requirements analysis and modelling techniques, especially in agent-oriented software engineering. Moreover goals are more useful and stable abstractions than others (e.g. user stories) in the analysis and design of software applications. Thus, the PRACTIONIST framework supports a goal-oriented approach for developing agent systems according to the Belief-Desire-Intention (BDI) model.**

In this paper we describe the goal model of PRACTIONIST agents, in terms of the general structure and the relations among goals. Furthermore we show how PRACTIONIST agents use their goal model to reason about goals during their deliberation process and means-ends reasoning as well as while performing their activities.

I. Introduction

With the increasing management complexity and maintenance cost of advanced information systems, attention in recent years has fallen on self-* systems and particularly on the autonomic computing approach and autonomic systems. In [1] authors argue that adopting a design approach that supports the definition of a space of possible behaviours related to the same function is one of the ways to make a system autonomic. Then the system should be able to select at runtime the best behaviour on the basis of the current situation. Goals can be used as an abstraction to model the functions around which the systems can autonomously select the proper behaviour.

In this view, the explicit representation of goals and the ability to reason about them play an important role in several requirements analysis and modelling techniques, especially when adopting the agent-oriented paradigm.

In this area, one of the most popular and successful agent models is the BDI [2], which derives from the philosophical tradition of practical reasoning first developed by Bratman [3]. It states that agents decide, moment by moment, which actions to perform in order to pursue their goals. Practical reasoning involves a deliberation process, to decide what states of affairs to achieve, and a means-ends reasoning, to decide how to achieve them.

Nevertheless there is a gap between BDI theories and several implementation [4]. Indeed, most of existing BDI agent platforms (e.g. JACK [5], JAM [6]) generally use goals instead of desires. Moreover, the actual implementations of mental states differ somewhat from their original semantics: desires (or goals) are treated as event types (such as in AgentSpeak(L) [7]) or procedures (such as in 3APL [8]) and intentions are executing plans. Therefore the deliberation process and means-ends reasoning are not well separated, as being committed to an intention (ends) is the same as executing a plan (means).

Moreover, some available BDI agent platforms do not support the explicit representation and implementation of goals or desires with their properties and relations, but they deal with them in a procedural and event-based fashion. As a result, while such an explicit representation of goals provide useful and stable abstractions when analysing and designing agent-based systems, there is a gap between the products of those phases and what development frameworks support.

According to Winikoff et al. [4], "by omitting the declarative aspect of goals the ability to reason about goals is lost". What is actually lost is the ability to *know* if goals are impossible, achieved, incompatible with other goals, and so forth. This in turn can support the *commitment strategies* of agents and their ability to autonomously drop, reconsider, replace or pursue goals.

However, some other BDI agent platforms deal with declarative goals. Indeed, in JADEX goals are explicitly represented according to a generic model, enabling the agents to handle their life cycle and reasoning about them [9]. Nevertheless, the model defined in JADEX does not deal with relations among goals.

The PRACTIONIST framework [10] 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. Indeed, in PRACTIONIST a goal is considered as an analysis, design, and implementation abstraction compliant to the semantics described in this paper. In other words, 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.

After a brief overview of the general structure of PRACTIONIST agents and their execution model (section II), this paper addresses the definition of the goal model (section III). We also describe how PRACTIONIST agents are able to reason about available goals according to their goal model,

current beliefs, desires, and intentions (see section IV). All aforementioned issues and the proposed model are fully implemented in the PRACTIONIST framework and available when developing applications by using the goal-oriented approach and the concepts described in this paper (section V). Finally, in section VI we present a simple example that illustrates the definition and the usage of goals and their relations.

II. PRACTIONIST AGENTS

The PRACTIONIST framework aims at supporting the programmer in developing BDI agents and is built on top of JADE [11], a widespread platform that implements the FIPA[1] specifications. Therefore, our agents are deployed within JADE containers and their main cycle is implemented by means of a JADE cyclic behaviour.

A PRACTIONIST agent is a software component endowed with the following elements:

- a set of *perceptions* and the corresponding *perceptors* that listen to some relevant external stimuli;
- a set of *beliefs* representing the information the agent has got about both its internal state and the external environment;
- a set of *goals* the agent wishes or wants to pursue. They represent some states of affairs to bring about or activities to perform and will be related to either its desires or intentions (see below);
- a set of *goal relations* the agent uses during the deliberation process and means-ends reasoning;
- a set of *plans* that are the means to achieve its intentions;
- a set of *actions* the agent can perform to act over its environment; and
- a set of *effectors* that actually execute the actions.

Beliefs, plans, and the execution model are briefly described in this section, while goals are the subject of this paper and are presented in the following sections. However, for a detailed description of the structure of PRACTIONIST agents, the reader should refer to [10].

The BDI model refers to beliefs instead of knowledge, as 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 false from the other agents' or the designer's point of view, but the idea is just to provide the agents with a subjective window over the world.

Therefore each PRACTIONIST agent is endowed with a prolog belief base, where beliefs are asserted, removed, or entailed through inference on the basis of KD45 modal logic rules [12] and user-defined formulas. Currently the PRACTIONIST framework supports two prolog engines, i.e. SWI-Prolog[2] and one that was derived from TuProlog[3].

In the PRACTIONIST framework plans represent an important container in which developers define the actual behaviors of agents.

[1]http://www.fipa.org
[2]http://www.swi-prolog.org
[3]http://tuprolog.alice.unibo.it

Each agent may own a declared set of plans (the *plan library*), each specifying the course of acts the agent will undertake in order to pursue its intentions, or to handle incoming perceptions, or to react to changes of its beliefs.

PRACTIONIST plans have a set of slots that are used by agents during the means-ends reasoning and the actual execution of agent activities. Some of these slots are: the trigger event, which defines the event (i.e. goals, perceptions, and belief updating) each plan is supposed to handle; the context, a set of condition that must hold before the plan can be actually performed; the body, which includes the acts the agent performs during the execution of the plan.

Through their perceptors, agents search for stimuli (perceptions) from the environment and transform them into (external) *events*, which in turn are put into the *Event Queue* (figure 1). Such a queue also contains internal events, which are generated when either an agent is committed to a goal or there is some belief updates. The former type of internal events is particularly important in PRACTIONIST agents, as described in the following sections.

The main cycle of a PRACTIONIST agent is implemented within a cyclic behaviour, which consists of the following steps.

1) it selects and extracts an event from the queue, according to a proper *Event Selection* logic;
2) it handles the selected event through the following *means-ends reasoning* process: (i) the agent figures out the *practical* plans, which are those plans whose trigger event matches the selected event (*Options* in figure 1); (ii) among practical plans, the agent detects the *applicable* ones, which are those plan whose context is believed true, and selects one of them (*main plan*); (iii) it builds the *intended means*, which will contain the main plan and other alternative practical plans. In case of goal event updates the corresponding intended means stack; otherwise it creates a new intended means stack.

It should be noted that every intended means stack can contain several intended means, each able to handle a given event, possibly through several alternative means.

Moreover all intended means stacks are concurrently executed, in order to provide the agents with the capability of performing several activities (perhaps referring to related or non-related objectives) in parallel. When executing each stack, the top level intended means is in turn executed, by performing its main plan. If it fails for some reason, one of alternative plans is then performed, until the corresponding ends (related to the triggering event) is achieved.

During the execution of a plan, several acts can be performed, such as *desiring* to bring about some states of affairs or to perform some action, *adding* or *removing* beliefs, *sending* ACL messages, and so forth. Particularly, desiring to pursue a goal triggers a deliberation/filtering process, in which the agent figures out whether that goal must be actually pursued or not, on the basis of the goal model declared for that agent.

The interaction among intended means belonging to different stacks can occur at a goal level, since each plan could wait

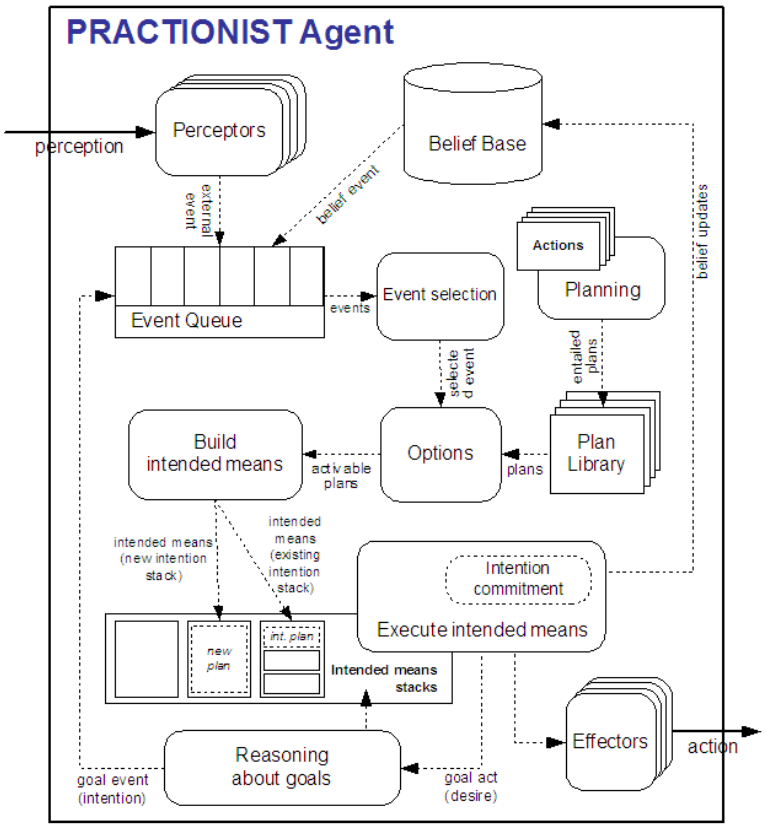

Fig. 1. PRACTIONIST Agent Architecture

for the success/failure of some goal that the agent is pursuing through another intended means.

III. Goal Model

In the PRACTIONIST framework, a goal is an objective to pursue and we use it as a mean to transform desires into intentions through the satisfaction of some properties. In other words, our agents are 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.

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

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

where:

- σ_g is the *success condition* of the goal g;
- π_g is the *possibility condition* of the goal g stating whether g can be achieved or not.

Since we consider such elements as local properties of goals, in the PRACTIONIST framework we defined them as operations that have to be implemented for each kind of goal (figure 3).

In order to describe the goal model, we first provide some definitions about the properties of goals.

Definition 1 A goal g_1 is *inconsistent* with a goal g_2 ($g_1 \perp g_2$) if and only if when g_1 succeeds, then g_2 fails.

Definition 2 A goal g_1 *entails* a goal g_2 or equivalently g_2 is *entailed by* g_1 ($g_1 \rightarrow g_2$) if and only if when g_1 succeeds, then also g_2 succeeds.

Definition 3 A goal g_1 is a *precondition* of a goal g_2 ($g_1 \mapsto g_2$) if and only if g_1 must succeed in order to be possible to pursue g_2.

Definition 4 A goal g_1 *depends* on a goal g_2 ($g_1 \hookrightarrow g_2$) if and only if g_2 is precondition of g_1 and g_2 must be successful while pursuing g_1.

Therefore the dependence is a stronger form of precondition. Both definitions let us specify that some goals must be successful before (and during, in case of dependency) pursuing some other goals (refer to section IV for more details).

Now, given a set G of goals and based on the above definitions, it is also possible to define some relations between those goals.

Definition 5 The inconsistency $\Gamma \subseteq G \times G$ is a binary symmetric relation on G, defining goals that are inconsistent with each other. Formally,

$$\Gamma = \{(g_i, g_j) \quad i,j = 1,...,|G| \quad :\ g_i \perp g_j\}. \tag{2}$$

When two goals are inconsistent with each other, it might

be useful to specify that one is preferred to the other. We denote that g_i is preferred to g_j with $g_i \succ g_j$.

Definition 6 The relation of preference $\Gamma^{'} \subseteq \Gamma$ defines the pair of goals (g_i, g_j) where $g_i \bot g_j$ and $g_i \succ g_j$. Formally,

$$\Gamma' = \{(g_i, g_j) \in \Gamma : g_i \succ g_j\}. \quad (3)$$

Therefore if there is no preference between two inconsistent goals, the corresponding pair does not belong to the set $\Gamma^{'}$. Moreover, since several goals can be pursued in parallel, there is no need to prefer some goal to another goal if they are not inconsistent each other.

Definition 7 The entailment $\Xi \subseteq G \times G$ is a binary relation on G, defining which goals entail other goals. Formally,

$$\Xi = \{(g_i, g_j) \quad i, j = 1, ..., |G| \ : \ g_i \rightarrow g_j\}. \quad (4)$$

Definition 8 The precondition set $\Pi \subseteq G \times G$ is a binary relation on G, defining which goals are precondition of other goals. Formally,

$$\Pi = \{(g_i, g_j) \quad i, j = 1, ..., |G| \ : \ g_i \mapsto g_j\}. \quad (5)$$

Definition 9 The dependence $\Delta \subseteq G \times G$ is a binary relation on G, defining which goals depend on other goals. Formally,

$$\Delta = \{(g_i, g_j) \quad i, j = 1, ..., |G| \ : \ g_i \hookrightarrow g_j\}. \quad (6)$$

Finally, on the basis of the above properties and relations we can now define the structure of the *goal model* of PRACTIONIST agents as follows

$$GM = \langle G, \ \Gamma, \ \Gamma', \ \Xi, \ \Pi, \ \Delta \rangle \quad (7)$$

where:

- G is the set of goals the agent could pursue;
- Γ is the *inconsistency* relation among goals;
- Γ' is the *preference* relation among inconsistent goals;
- Ξ is the *entailment* relation among goals;
- Π is the *precondition* relation among goals;
- Δ is the *dependence* relation among goals.

IV. Reasoning about goals

In this section we show how the goal elements previously defined are used by PRACTIONIST agents when reasoning about goals during their deliberation process and the means-ends reasoning. We also highlight the actual relations between them and mental attitudes, i.e. desires and intentions.

In PRACTIONIST agents goals and their properties are defined on the basis of what agents believe. Thus, an agent will believe that a goal $g = \langle \sigma_g, \ \pi_g \rangle$ has succeeded if it believes that its success condition σ_g is true. The same holds for the other properties.

It is important to note that, in PRACTIONIST, desires and intentions are mental attitudes towards goals, which are in turn considered as descriptions of objectives. Thus, referring to a goal, an agent can just relate it to a *desire*, which it is not committed to because of several possible reasons (e.g. it believes that the goal is not possible). On the other hand, a goal can be related to an *intention*, that is the agent is actually and actively committed to pursue it.

Let $GM = \langle G, \ \Gamma, \ \Gamma', \ \Xi, \ \Pi, \ \Delta \rangle$ be a *goal model* of a PRACTIONIST agent α and, at a given time, $G' \subseteq G$ be the set of its active goals, which are those goals that the agent is already committed to.

Suppose that α starts its deliberation process and generates the goal $g = \langle \sigma_g, \ \pi_g \rangle$ as an option. Therefore the agent would like to commit to g, that is its *desire* is to bring about the goal g. However, since an agent will not be able to achieve all its desires, it performs the following process in the context of its deliberation phase (figure 2): the agent checks if it believes that the goal g is *possible* and not *inconsistent* (see definition 1) with active goals (belonging to G').

If both conditions hold the desire to pursue g will be promoted to an *intention*. Otherwise, in case of inconsistency among g and some active goals, the desire to pursue g will become an intention only if g is preferred to such inconsistent goals, which will in turn be dropped.

In any case, if the desire to pursue g is promoted to an *intention*, before starting the means-ends reasoning, the agent α checks if it believes that the goal g *succeeds* (that is, if it believes that the success condition σ_g holds) or whether the goal g is entailed (see definition 2) by some of the current active goals. In case of both above conditions do not hold, the agent will perform the means-ends reasoning, by either selecting a plan from a fixed plan library or dynamically generating a plan and finally executing it (details on this means-ends reasoning can be found in [10]).

Indeed, if the goal g succeeds or is entailed by some current active goals (i.e. some other means is working to achieve a goal that entails the goal g), there is no reason to pursue it. Therefore, the agent does not need to make any means-ends reasoning to figure out how to pursue the goal g.

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

In the PRACTIONIST framework, as a default, an agent 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. This commitment strategy to intention is called *single-minded commitment* [13]. In order to perform such a behaviour, the agent continuously checks if it believes that the goal g has just succeeded and that the goal g is still possible.

Moreover the agent checks if some dependee goal does not succeed. If so, it will desire to pursue such a goal and then continue pursuing the goal g. When all dependee goals succeed, the agent resumes the execution of the plan.

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

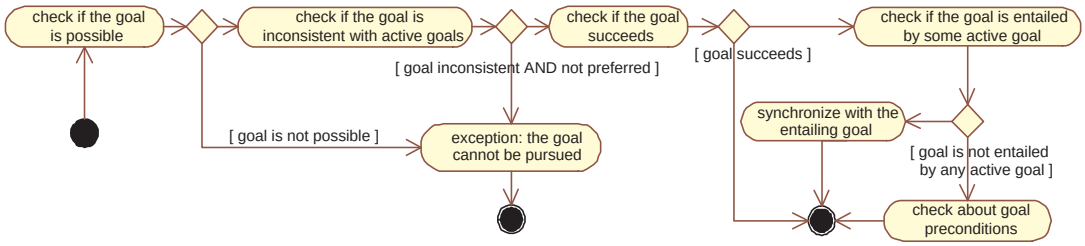

Fig. 2. Reasoning about goals: the deliberation phase.

same intended means and executes it.

If none of the alternative plans was able to successfully pursue the goal g, the agent take into consideration the goals that *entail* g. Thus the agent selects one of them and considers it as an option, processing it in the way described in this section, 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).

V. The Support for the Goal Model in the PRACTIONIST Framework

In order to provide the PRACTIONIST framework with the support for the definition/handling of agent goal models and the capabilities for reasoning about goals, we identified and fulfilled the following requirements:

- registration of the goals that each agent could try to pursue during his life cycle;
- registration of the relations among such goals;
- checking whether two goals are inconsistent and which the preferred one is (if any);
- getting the list of goals that entail a given goal;
- getting the list of goals that are precondition of a given goal;
- getting the list of goals which a given goal depends on.

A proper ad-hoc search algorithm explores the goal model and answers the queries, on the basis of both declared and implicit relations. Indeed, implicit relations (especially inconsistence and entailment) can be inferred from the semantics of some built-in goals, such as state goals (e.g. $achieve(\varphi)$, $cease(\varphi)$, $maintain(\varphi)$, and $avoid(\varphi)$, where φ is a closed formula of FOL). Therefore, the goal reasoner takes into account implicit relations such as $achieve(\varphi) \perp achieve(\neg\varphi)$, $achieve(\varphi) \perp cease(\varphi)$, $maintain(\varphi) \perp avoid(\varphi)$, and so forth.

Figure 3 shows the actual structure of the `GoalModel` that each agent owns (`PRACTIONISTAgent` is the abstract class that has to be extended when developing PRACTIONIST agents). Such a model stores information about declared goals (with their internal properties, i.e. success and possibility condition) and the four types of relations these goals are involved in. Specifically the interface `GoalRelation` provides the super interface for all goal relations supported by the PRACTIONIST framework (i.e. `EntailmentRel`, `InconsistencyRel`, `DependencyRel`, and `PreconditionRel`) and defines the operation `verifyRel`, whose purpose is to check each specific relation.

In order to exploit the features provided by the goal model and understand if a given goal the agent desires to pursue is inconsistent with or implied by some active goals, the agent must have information about such active goals and whether them are related to either desires or intentions. Therefore, each PRACTIONIST agent owns an `ActiveGoalsHandler` component, which, with the aid of the `GoalModel`, has the responsibility of keeping track of all executing intended means stacks with the corresponding waiting and executing goals and managing requests made by the agent.

Thus, at any given time, the `ActiveGoalsHandler` is aware of current desires and intentions of the agent, referring them to active goals.

VI. An Example

In this section we present the Tileworld example to illustrate how to use the goal model presented in this paper and the support provided by the PRACTIONIST framework.

The Tileworld example was initially introduced in [14] as a system with a highly parameterized environment that could be used to investigate the reasoning in agents. The original Tileworld consists of a grid of cells on which tiles, obstacles and holes (of different size and point value) can exist. Each agent can move up, down left or right within the grid to pick up and move tiles in order to fill the holes. Each hole has an associated score, which is awarded to the agent that has filled the hole. The main goal of the agent is to score as many points as possible.

Tileworld simulations are dynamic and the environment changes continually over time. Since this environment is highly parameterized, the experimenter can alter various aspects of it through a set of available ”knobs”, such as the rate at which new holes appear (*dynamism*), the rate at which obstacles appear (*hostility*), difference in hole scores (*variability of utility*), and so forth.

Such applications, with a potentially high degree of dynamism, can benefit from the adoption of a goal-oriented design approach, where the abstraction of goal is used to declaratively represent agents' objectives and states of affairs that can be dynamically achieved through some means.

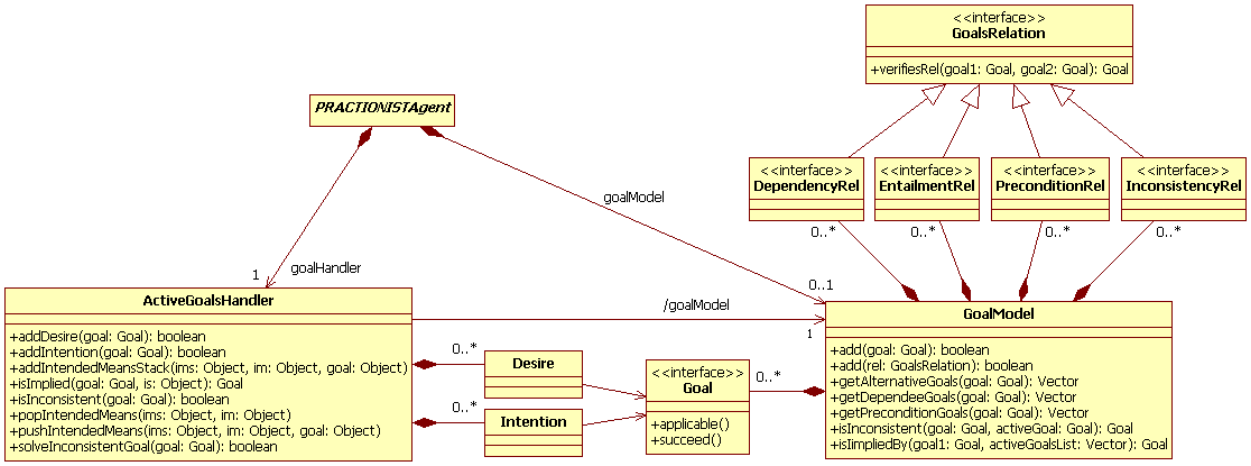

Fig. 3. The structure of the support for the goal model in the PRACTIONIST framework.

Figure 4 shows the Tileworld environment, where new agents can be added or removed and the corresponding parameters can be dynamically changed.

In our Tileworld demonstrator two types of agents were developed, the Tileworld Management Agent (TWMA) and the Tileworld Player Agent (TWPA): the former is the agent that manages and controls the environment, by creating and destroying tiles, holes and obstacles, according to the parameters set by the user; the latter is the agent moving within the grid and whose primary goal is to maximize its score by filling holes with tiles. A player agent does not get any notification about the environment changes (i.e. by the management agent), but it can ask such an information (e.g. what the current state of a cell is) by means of sensing actions, in order to adopt the best strategy on the basis of the current state of the environment. In fact, for each state of the environment (e.g., static, dynamic, very dynamic, etc.) at least a strategy is provided. All the strategies are implemented through plans that share the same goal and differ for their operative conditions (i.e. the context).

It should be noted that, since PRACTIONIST agents are endowed with the ability of dynamically building plans starting from a given goal and a set of available actions, some strategies could be generated on-the-fly by taking into account emerging situations.

The player agent has beliefs about the objects that are placed into the grid, its position, its score, the state of the environment, etc.

The TWPA top level goal is to score as many points as possible, but to do this, it has to register itself with the manager, look for the holes and for the tiles, hold a tile, and fill a hole.

We designed the TWPA by adopting the goal-oriented approach described in this paper and directly implemented its goal-related entities (i.e. goals and relations) thank to the support provided by the PRACTIONIST framework. In figure 5 a fragment of the goal model of the TWPA is shown as a UML class diagram with dependencies stereotyped with the name of the goal relations. Actually some relations only hold under certain condition and the diagram does not show such details.

According to the diagram, the TWPA has to be registered with the TWMA before increasing its score (the goal `ScorePoints` *depends on* the goal `RegisterWithManager`). Moreover, in order to score points, the TWPA has to fill as many holes as possible (the goal `FillHole` *entails* the goal `ScorePoints`). But, in order to fill a hole, the TWPA has to hold a tile and find a hole (the goal `FillHole` *depends on* the goal `HoldTile` and requires the goal `FillHole` as *precondition*); finally, the TWPA has to find the tile to hold it (the goal `HoldTile` has the goal `FindTile` as a precondition).

According to the above-mentioned description, the following source code from the TWPAgent class shows how goals and relations among them are added to the agent and thus how to create the goal model through the PRACTIONIST framework:

```
protected void initialize()
{
 ...
 GoalModel gm = getGoalModel();

 // Goal declaration
 gm.add(new RegisterWithManager());
 gm.add(new ScorePoints());
 gm.add(new HoldTile());
 gm.add(new FindTile());
 gm.add(new FillHole(getBeliefBase()));
 gm.add(new FindHole());

 // relations among goals
 gm.add(new Dep_ScorePoints_RegisterWithManager());
 gm.add(new Ent_ScorePoints_FillHole());
 gm.add(new Dep_FillHole_HoldTile());
 gm.add(new Pre_HoldTile_FindTile());
 gm.add(new Pre_FillHole_FindHole());

 ...
}
```

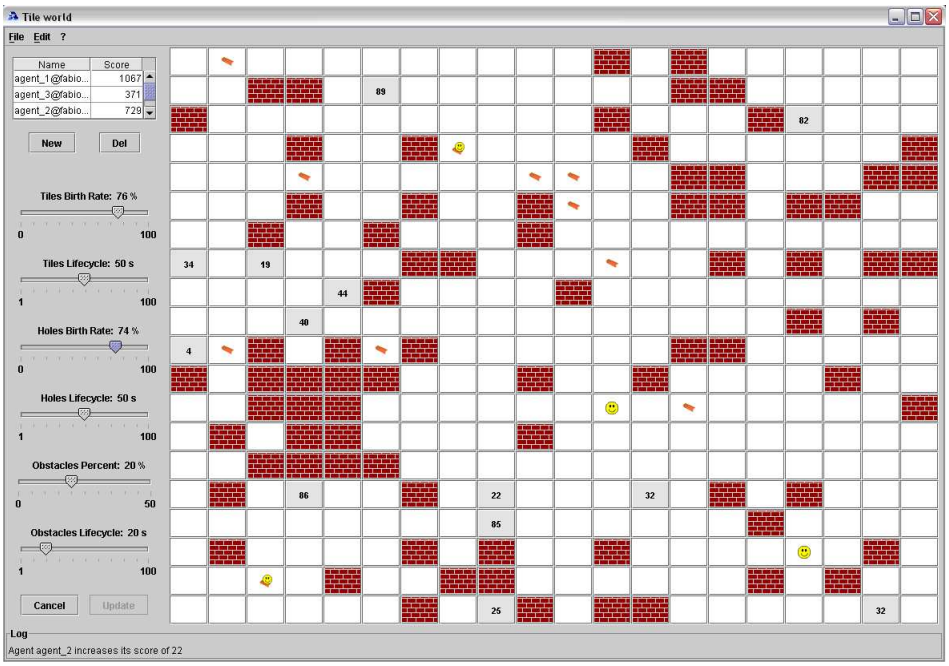

Fig. 4. The Tileworld environment.

In order to better understand how the above-mentioned relations are implemented, the following source code shows the precondition relation among the goals `HoldTile` and `FindTile`:

```
public class Pre_HoldTile_FindTile
      implements PreconditionRel
{
  public Goal verifyRel(Goal goal1, Goal goal2)
   {
    if((goal1 instanceof HoldTile) &&
        (goal2 instanceof FindTile))
           return new FindTile;

     return null;
  }
  ...
}
```

When the player agent desires to pursue a goal, it checks if this goal is involved in some relations and in that case it reasons about them during the deliberation, means-ends, and intention reconsideration processes. Thus, developers only need to specify goals and relations among them at the design time.

As an example, when the TWPA desires to fill a hole (i.e. `FillHole`), according to the defined goal model and the semantics described in section 2, the agent automatically will check if it just holds a tile (i.e. `HoldTile`); if not, such a goal will be desired. On the other hand, the agent will check if it has found a hole (i.e. `FindHole`) and again, if not, it will desire that.

Moreover, when pursuing the goal `FillHole`, the agent will continuously check the success of all its dependee goals (i.e. `HoldTile`) and *maintain* them in case of failure.

It should be noted that the plan to pursue the goal `FillHole` does not need to include the statements to desire either the dependee (i.e. `HoldTile`) or precondition (i.e. `FindHole`) goals, as shown in the following code fragment.

```
public class FillHolePlan extends GoalPlan
{
  public void body() throws PlanExecutionException
  {
    String posPred = "pos(obj1: X,obj2: Y)";
    AbsPredicate pos =
      getBeliefBase().retrieveAbsPredicate(
        AbsPredicateFactory.create(posPred));

    int xPos = pos.getInteger("obj1");
    int yPos = pos.getInteger("obj2");

    doAction(new ReleaseTileAction(xPos, yPos,
       twaServer.getHoleValue(xPos, yPos)));
    ...
    }
  ...
}
```

The Tileworld domain highlights how the PRACTIONIST goal model is particularly adequate to model dynamic environments in a very declarative manner.

VII. CONCLUSIONS AND FUTURE WORK

In the PRACTIONIST framework, desires and intentions are mental attitudes towards goals, which are in turn considered as descriptions of objectives.

In this paper we described how a declarative representation of goals can support the definition of desires and intentions in PRACTIONIST agents. It also supports the detection and the resolution of conflicts among agents' objectives and activities. This results in a reduction of the gap between BDI theories and several available implementations.

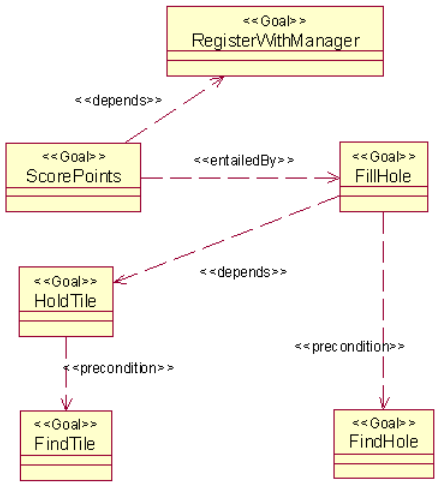

Fig. 5. TWPA's goal model.

We also described how goals and relations are used by PRACTIONIST agents during their deliberation process and the execution of their activities; particularly it is described how agents manages these activities by using the support for the goal model shown in the previous sections.

It should be noted that, unlike several BDI and non-BDI agent platforms, the PRACTIONIST framework supports the declarative definition of goals and the relations among them, as described in this paper. This provides the ability to *believe* if goals are impossible, already achieved, incompatible with other goals, and so forth. This in turn supports the *commitment strategies* of agents and their ability to autonomously drop, reconsider, replace or pursue intentions related to active goals.

The ability of PRACTIONIST agents to reason about goals and the relations among them (as described in section IV) lets programmers implicitly specify several behaviours for several circumstances, without having to explicitly code such behaviours, letting agents figure out the right activity to perform on the basis of the current state and the relations among its potential objectives.

Goals can be adopted throughout the whole development process. Thus, we are defining a development methodology where goals play a central role and maintain the same semantics from early requirements to the implementation phase.

As a part of our future strategy, we aims at extending the proposed model with further properties of goals and relations among them. Finally, we aim at applying the concepts and the model described in this paper in the development of real-world applications based on BDI agents.

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

REFERENCES

[1] A. Lapouchnian, S. Liaskos, J. Mylopolous, and Y. Yu, "Towards requirements-driven autonomic systems design," *Proceedings of the 2005 workshop on Design and evolution of autonomic application software*, pp. 1–7, 2005, aCM Press, New York, NY, USA.

[2] A. S. Rao and M. P. Georgeff, "BDI agents: from theory to practice," in *Proceedings of the First International Conference on Multi—Agent Systems*. San Francisco, CA: MIT Press, 1995, pp. 312–319. [Online]. Available: http://www.uni-koblenz.de/~fruit/LITERATURE/rg95.ps.gz

[3] M. E. Bratman, *Intention, Plans, and Practical Reason*. Cambridge, MA: Harvard University Press, 1987.

[4] M. Winikoff, L. Padgham, J. Harland, and J. Thangarajah, "Declarative & procedural goals in intelligent agent systems," in *KR*, 2002, pp. 470–481.

[5] P. Busetta, R. Rnnquist, A. Hodgson, and A. Lucas, "Jack intelligent agents - components for intelligent agents in java," 1999.

[6] M. J. Huber, "Jam: A bdi-theoretic mobile agent architecture." in *Agents*, 1999, pp. 236–243.

[7] A. S. Rao, "AgentSpeak(L): BDI agents speak out in a logical computable language," in *Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, R. van Hoe, Ed., Eindhoven, The Netherlands, 1996. [Online]. Available: citeseer.ist.psu.edu/article/rao96agentspeakl.html

[8] K. V. Hindriks, F. S. D. Boer, H. W. van der, and J. J. Meyer, "Agent programming in 3APL," *Autonomous Agents and Multi-Agent Systems*, vol. 2, no. 4, pp. 357–401, 1999, publisher: Kluwer Academic Publishers, Netherlands.

[9] L. Braubach, A. Pokahr, W. Lamersdorf, and D. Moldt, "Goal representation for bdi agent systems," in *Second International Workshop on Programming Multiagent Systems: Languages and Tools*, 7 2004, pp. 9–20.

[10] V. Morreale, S. Bonura, G. Francaviglia, M. Cossentino, and S. Gaglio, "Practionist: a new framework for bdi agents," in *Proceedings of the Third European Workshop on Multi-Agent Systems (EUMAS'05)*, 2005, p. 236.

[11] F. Bellifemine, A. Poggi, and G. Rimassa, "JADE - a FIPA-compliant agent framework," in *Proceedings of the Practical Applications of Intelligent Agents*, 1999. [Online]. Available: http://jmvidal.cse.sc.edu/library/jade.pdf

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

[13] 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

[14] M. E. Pollack and M. Ringuette, "Introducing the tileworld: Experimentally evaluating agent architectures," *National Conference on Artificial Intelligence*, pp. 183 – 189, 1990.