Goal-Oriented Development of BDI Agents: the PRACTIONIST Approach

Vito Morreale, Susanna Bonura, Giuseppe Francaviglia, Fabio Centineo
R&D Lab - ENGINEERING Ingegneria Informatica S.p.A.
Viale R. Siciliana, 7275 - Palermo - ITALY
{morreale, bonura, giuseppe.francaviglia, fabio.centineo}@eng.it

Massimo Cossentino
ICAR - Italian National Research Council
Viale delle Scienze - Palermo - ITALY
cossentino@pa.icar.cnr.it

Salvatore Gaglio
DINFO - University of Palermo
Viale delle Scienze - Palermo - ITALY
gaglio@unipa.it

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, as goals are more stable than other abstractions (e.g. user stories).

In PRACTIONIST, a framework for developing agent systems according to the Belief-Desire-Intention (BDI) model, goals play a central role. Thus, in this paper we describe the structure of the goal model in the PRACTIONIST framework and how agents use their goal model to reason about goals, desires, and intentions during their deliberation process and means-ends reasoning as well as while performing their activities.

1 Introduction

With the increasing management complexity and maintenance cost of advanced information systems, in recent years attention has been turned towards self-* systems and particularly the autonomic computing approach and autonomic systems. In [6] authors argue that one of the ways to make a system autonomic concerns with the adoption of a design approach that supports the definition of a space of possible behaviours related to the same function. 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. The abstraction of goal could be particularly useful and appropriate, especially when adopting the agent-oriented paradigm, which provides interesting abstractions related to autonomous entities for the development of software systems whose requirements are not entirely known at design time (e.g. when running in rapidly changing environments).

One of the most popular and successful agent models is the Belief-Desire-Intention (BDI) [9], which derives from the philosophical tradition of practical reasoning first developed by Bratman [1]. 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.

Most of existing BDI agent platforms (e.g. JACK [3], JAM [5]) generally use goals instead of desires. Moreover, the actual implementations of mental states differ from their original semantics: desires (or goals) are treated as event types (such as in AgentSpeak(L) [8]) or procedures (such as in 3APL [4]) 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). As a result, there is a gap between BDI theories and implementations [10].

Moreover, some available BDI agent platforms do not support the explicit representation and implementation of goals or desires with their properties and relations. As a result, while such an explicit representation of goals provides 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.

Actually, several authors have argued the importance of declarative representations of goals in agent deliberation processes, especially in dynamic environments. Among them, Winikoff et al. [10] stated that ”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 supports the *commitment strategies* of agents and the capability to autonomously drop, reconsider, replace or pursue goals.

However, some other BDI agent platforms deal with declarative goals. Indeed, in JADEX agent platform, goals are explicitly represented according to a generic model, enabling the agents to handle their lifecycle and reasoning about them [2]. Nevertheless, the model defined in JADEX does not deal with relations among goals.

Our PRACTIONIST framework [7] 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. In PRACTIONIST a goal is considered as an analysis, design, and implementation abstraction compliant to the semantics described below. 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.

This paper, after a brief overview of the general structure of PRACTIONIST agents and their execution model (section 2), addresses the definition of the goal model (section 3). 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 4). Section 5 describes how the aforementioned concepts are implemented in the PRACTIONIST framework, while in section 6 we present a simple example that illustrates how to define and use goals and their relations.

2 The PRACTIONIST framework

The PRACTIONIST framework supports programmers in developing BDI agents *(i)* endowed with a symbolic representation about their internal states and the external environment, *(ii)* able to plan their activities in order to pursue some objectives, and *(iii)* provided with the ability of both proactively and reactively performing activities. We chose to define our framework on top of JADE [1].

A PRACTIONIST agent is a software component endowed with the following elements: a set of *perceptors* that listen to some relevant external stimuli (*perceptions*); 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, which represent some states of affairs to bring about or activities to perform and can be related to either *desires* or *intentions*; 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 the 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. For a detailed description of PRACTIONIST agents, refer to [7].

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 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 behaviours and strategies of agents. 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, which are used by agents during the means-ends reasoning and the actual execution of their 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 conditions that must be believed true before performing the plan; the *body*, which include the acts the agent performs during the execution of the plan. Within the body several acts are possible, such as *sending* messages, *desiring* to bring about some states of affairs or perform some action, modifying beliefs, and so forth.

The agent main cycle is implemented within a cyclic behaviour, in which the following steps are executed (fig. 1):

1. through the perceptors, it searches for perceptions from the environment and transforms them into (external) *events*, which in turn are put into the *Event Queue*;
2. it selects and extracts an event from the queue, according to a proper *Event selection* logic;
3. 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 the other alternative practical plans, and updates the intended means stack set (i.e. in case of goal event the new intended means is put on top of an existing stack, otherwise a new stack is created).

[1] http://jade.tilab.com

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

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

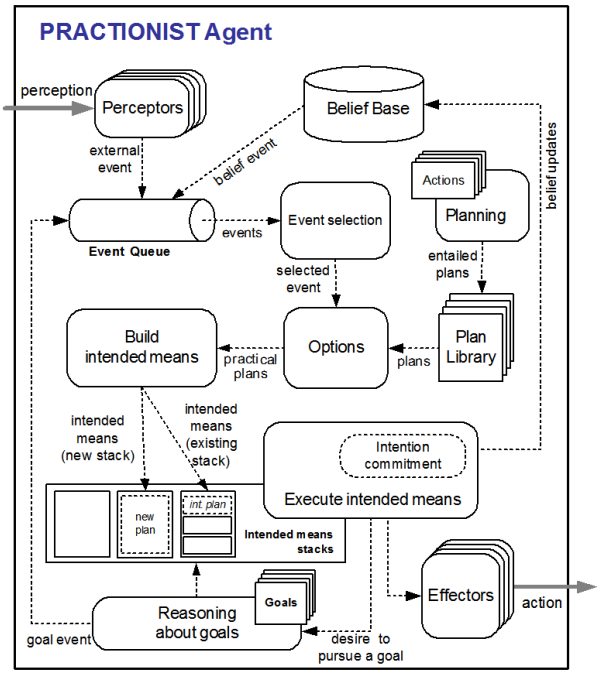

Figure 1. PRACTIONIST Agent Architecture.

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.

3 The goal model

In the PRACTIONIST framework a *goal* is a description of an objective to pursue and an abstraction to make the distinction between the state of affairs to be achieved and the way to achieve it. Besides, we use goals as a mean to transform desires into intentions through the satisfaction of some properties. Therefore PRACTIONIST 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.

In this section we provide a model of PRACTIONIST goals, in terms of their success conditions and other properties, such as possibility, inconsistency with other goals, entailment between goals, and so forth. 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 whether the agent should give up to pursue the goal g 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 (as shown in figure 2 in the interface `Goal`).

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 a precondition of g_1 and g_2 must be successful while pursuing g_1.

Therefore the dependency is a stronger form of precondition. Now, given a set G of goals and on the basis of the previous definitions, it is also possible to define some relations between those goals.

Definition 5. The binary symmetric relation $\Gamma \subseteq G \times G$ defines pairs of inconsistent goals that belong to G. Formally, $\Gamma = \{(g_i, g_j) \ \ i, j = 1, ..., |G| \ : \ g_i \perp g_j\}$.

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 $g_i \succ g_j$. It should be noted that, since in PRACTIONIST 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 6. The relation $\Gamma' \subseteq \Gamma$ defines the pair of goals (g_i, g_j) such that $g_i \perp g_j$ and $g_i \succ g_j$. Formally, $\Gamma' = \{(g_i, g_j) \in \Gamma : g_i \succ g_j\}$.

Therefore if there is no preference between two inconsistent goals, the corresponding pair does not belong to Γ'.

Definition 7. The binary relation $\Xi \subseteq G \times G$ defines which goals entail other goals. Formally, $\Xi = \{(g_i, g_j) \ \ i, j = 1, ..., |G| \ : \ g_i \rightarrow g_j\}$.

Definition 8. The binary relation $\Pi \subseteq G \times G$ defines which goals are precondition of other goals. Formally, $\Pi = \{(g_i, g_j) \ \ i, j = 1, ..., |G| \ : \ g_i \mapsto g_j\}$.

Definition 9. The binary relation $\Delta \subseteq G \times G$ defines which goals depend on other goals. Formally, $\Delta = \{(g_i, g_j) \;\; i, j = 1, ..., |G| \; : \; g_i \hookrightarrow g_j\}$.

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

$$GM = \langle G,\ \Gamma,\ \Gamma',\ \Xi,\ \Pi,\ \Delta \rangle\,, \tag{2}$$

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, and Δ is the *dependence* relation among goals.

4 Reasoning about goals

PRACTIONIST agents use the goal elements previously defined during their deliberation process and the means-ends reasoning. In this section we present the actual relationships between goals and mental attitudes.

As already mentioned, since PRACTIONIST agents are compliant to the BDI model, goals and their properties are defined according to what agents believe. Thus, as an informal example, an agent will believe that a goal has succeeded if it believes that its success condition is true. The same holds for the other properties.

It is important to note that in our view desires and intentions are mental attitudes towards goals. Thus an agent can just relate a goal 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*, when the agent is actually and actively committed to pursue it.

Thus, given a goal model $GM = \langle G,\ \Gamma,\ \Gamma',\ \Xi,\ \Pi,\ \Delta \rangle$ and fixed a goal g belonging to the set G, it is possible to figure out the set Γ_g of goals that are *inconsistent* with g, the set Γ'_g of goals that are *inconsistent* with and *preferred* to g, the set Γ''_g of goals *inconsistent* with g, which in turn is *preferred* to them, the set Ξ_g of goals that *entail* g, the set Π_g of goals that are *precondition* for g, and the set Δ_g of goals which g *depends* on. Formally,

$$\Gamma_g = \{g_i,\ i = 1, ..., |G| : (g_i, g) \in \Gamma\}, \tag{3}$$

$$\Gamma'_g = \{g_i,\ i = 1, ..., |G| : (g_i, g) \in \Gamma'\}, \tag{4}$$

$$\Gamma''_g = \{g_i,\ i = 1, ..., |G| : (g, g_i) \in \Gamma'\}, \tag{5}$$

$$\Xi_g = \{g_i,\ i = 1, ..., |G| : g_i \rightarrow g\}, \tag{6}$$

$$\Pi_g = \{g_i,\ i = 1, ..., |G| : g_i \mapsto g\}, \tag{7}$$

$$\Delta_g = \{g_i,\ i = 1, ..., |G| : g \hookrightarrow g_i\}. \tag{8}$$

For any goal g, some elements of Γ_g might not belong to either Γ'_g or Γ''_g. Formally, $\forall g \in G,\ \Gamma'_g \cup \Gamma''_g \subseteq \Gamma_g$. Moreover, no element of Γ_g can belong to both Γ'_g and Γ''_g, that is $\Gamma'_g \cap \Gamma''_g = \oslash$. Finally, it turns out that $\Delta_g \subseteq \Pi_g$.

Thus, let GM be the goal model of a PRACTIONIST agent α and, at a given time, $G' \subseteq G$ be the set of goals which the agent is already committed to (i.e. *active goals*).

Suppose that α starts its deliberation process and generates a goal g as an option, that is it would like to commit to g. Therefore its *desire* is to bring about the goal g. However, since any agent will not be able to achieve all its desires, α performs the following process in the context of its deliberation phase: it checks if it believes that the goal g is *possible* and not *inconsistent* with active goals (belonging to G'). The goal g is not inconsistent with current active goals if and only if $G' \cap \Gamma_g = \varnothing$.

If both conditions hold the desire to pursue g will be promoted to an *intention*. On the other hand, in case of inconsistency among g and some active goals,

1. if the goal g is preferred to all active goals that are inconsistent with it (formally, $G' \cap \Gamma_g = G' \cap \Gamma''_g$), the agent will give up pursuing them and will drop the corresponding intentions and the desire related to g will become a new intention;

2. otherwise, if either the goal g is not preferred to all active goals inconsistent with it ($G' \cap \Gamma_g \neq G' \cap \Gamma''_g$), or some of active goals inconsistent with g is preferred to it ($G' \cap \Gamma'_g \neq \oslash$), or in case of no preferences ($G' \cap \Gamma'_g = G' \cap \Gamma''_g = \oslash$), the agent will prefer its current intentions and drop g (which will just remain related to a desire).

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* or whether the goal g is entailed by some of the current active goals. The goal g is entailed by some current active goal if and only if $G' \cap \Xi_g \neq \oslash$. 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, α does not need to make any means-ends reasoning to figure out how to pursue g.

In case of above conditions do not hold, the agent can perform the means-ends reasoning, by either selecting a plan from the plan library or dynamically generating a plan and finally executing it (details on this can be found in [7]). However, before then, if some declared goals are *preconditions* for g, that is $\Pi_g \neq \oslash$, the agent will first desire to pursue such goals and then the goal g.

Related to the issue of reasoning about goals are intention commitment strategies, which concern with the reconsideration ability of agents. In PRACTIONIST, 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 it. This commitment strategy to intention is called *single-minded commitment*.

In order to behave according to such a commitment strategy, during the execution of the intended means built to bring about the success of a given goal g, the agent continuously checks *(i)* if it believes that the goal g has just succeeded, *(ii)* if it believes that the goal g is still possible, and *(iii)* if it believes that the goal should be cancelled for any other reason. Meanwhile the agent also checks if some dependee goals do not succeed. If so, it will desire to pursue such goals and then continue pursuing the goal g.

Each PRACTIONIST agent is also able to recover from *plan failures* and try other means to achieve an intention, when the selected plan fails or is no longer appropriate. Thus, it selects one of applicable *alternative plans* within the same intended means and executes it.

If none of the alternative plans was able to successfully pursue the goal g, the agent takes into consideration other goals that *entail* g. Thus it 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 there is no way to pursue that intention. Thus, according to the beliefs, the goal was *possible*, but the agent was not able to pursue it.

5 The goal model in the PRACTIONIST framework

Within the PRACTIONIST framework we included the support for the definition/handling of agent goal models and the capabilities for reasoning about goals. Specifically the framework provided facilities for the registration of the goals that each agent could try to pursue during his life cycle, the 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 or are precondition of a given goal, getting the list of goals which a given goal depends on.

Some of the above features affect the design activity, while others are exploited by the agent in a transparent way during its life cycle. The goal registration features are directly used by agent developers, who define goals and relations for each agent before inserting them into the goal model during the agent initialization phase (see the goal model of the example reported in section 6).

In order to fulfil the above requirements, a proper ad-hoc search algorithm explores the goal model and answers the queries, on the basis of existing relations. Moreover, implicit relations (especially inconsistence and entailment) can be inferred from the semantics of some built-in 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 also takes into account implicit relations such as $achieve(\varphi) \perp achieve(\neg\varphi)$, $achieve(\varphi) \perp cease(\varphi)$, $maintain(\varphi) \perp avoid(\varphi)$, $maintain(\varphi) \rightarrow achieve(\varphi)$, and so forth.

Figure 2 shows the actual structure of the `GoalModel` that each agent owns (`PRACTIONISTAgent` is the abstract class to extend when developing PRACTIONIST agents). Such a model stores information about declared goals (with their success, possibility, and cancel conditions) and the relations these goals are involved in. `GoalRelation` is the super interface for all goal relations supported by the framework (i.e. `EntailmentRel`, `InconsistencyRel`, `DependencyRel`, and `PreconditionRel`) and defines the operation `verifyRel` to check each relation.

In order to exploit the features provided by the goal model and understand if a given goal that the agent desires to pursue is inconsistent with or implied by some active goals, the agent itself 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. Thus, at any given time, the `ActiveGoalsHandler` is aware of the current desires and intentions of the agent.

6 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 introduced to simulate highly parameterized environments and test the meta-level reasoning of agents. The environment consists of a grid of cells on which tiles, obstacles and holes (of different size and value) can exist. Every 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 can continually change over time, as several parameters can be manually modified. Such an application 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.

In our Tileworld demonstrator two types of agents were developed, i.e. the Tileworld Management Agent (TWMA)

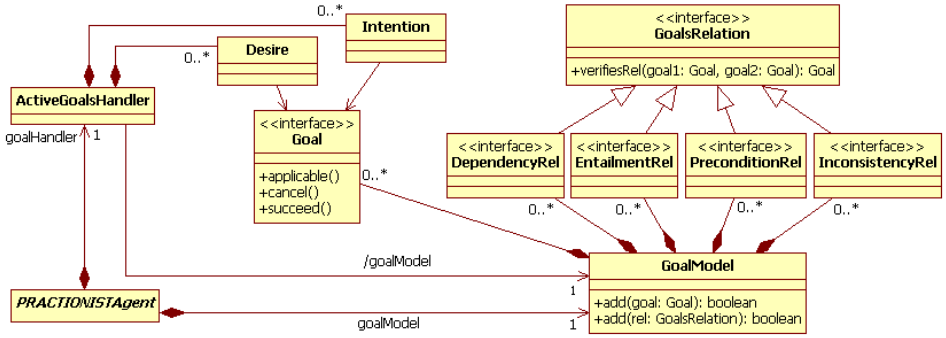

Figure 2. The structure of the support for the goal model in the PRACTIONIST framework.

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 aims at maximizing its score by filling holes with tiles.

The TWPA adopts the best strategy on the basis of the current state of the environment (e.g., static, dynamic, very dynamic, etc.). All strategies are implemented through plans that share the same goal and differ for their operative conditions (i.e. the context).

It should be also 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 to face 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. 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 3 a fragment of the goal model of the TWPA is shown as a UML class diagram where dependencies are stereotyped with the name of the goal relations. Actually some relations only hold under certain conditions 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 to find a hole (the goal `FillHole` *depends on* the goal `HoldTile` and requires the goal `FillHole` as a *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 to create the goal model in terms of goals and relations among them, which are added to the agent:

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

 // Goal declaration
 gm.add(new RegisterWithManager());
 ...
 gm.add(new ScorePoints());

 // Relations among goals
 gm.add(new Dep_ScorePoints_RegisterWithMan());
 gm.add(new Ent_ScorePoints_FillHole());
 ...
 gm.add(new Pre_FillHole_FindHole());
 ...
}
```

In order to better understand how the above-mentioned relations are implemented, the following source code shows a simple implementation of the dependency relation among the goals `FillHole` and `HoldTile`:

```
class Dep_FillH_HoldT implements DependencyRel {
  public Goal dependsOn(Goal g1, Goal g2) {
    if((g1 instanceof FillHole) &&
      (g2 instanceof HoldTile)) {
        return g2;
      }
      return null;
   }
}
```

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

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 4, it will automatically check

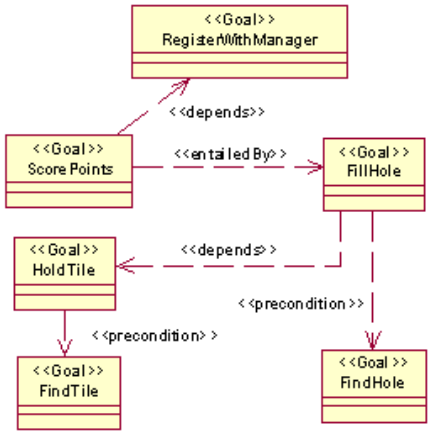

Figure 3. TWPA's goal model.

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 goals which `FillHole` depends on (i.e. `HoldTile`) and *maintain* them in case of failure. Therefore the plans to pursue any goal g do not need to include the statements to desire either its dependee or precondition goals.

7 Conclusions and future work

In our framework, desires and intentions are mental attitudes towards goals, which are in turn considered as descriptions of objectives. In this paper we presented how a declarative representation of goals can support the definition of desires and intentions in PRACTIONIST agents. Such an approach also supports the detection and the resolution of conflicts among agents' objectives and activities.

Thus, we defined the structure of goals and some relations among them (i.e. *inconsistency*, *entailment*, *dependence*, *precondition*). Then we described how goals and relations are used by PRACTIONIST agents during their deliberation process and the execution of their activities.

The main contribution of our work is that, unlike several BDI and non-BDI agent platforms, the PRACTIONIST framework supports the declarative definition of goals and the relations among them. This provides the ability to *believe* if goals are impossible, already achieved, incompatible with other goals, etc. and 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 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. As an informal result, the code of the Tileworld with the goal model is 30% less than the one without the goal model. As a part of our future strategy, we aim at evaluating more scientifically this potential benefit.

Obviously, the usage of the goal model at run-time is a little time-consuming and decrease the performance of agents. However, programmers can disable some of the capabilities discussed in this paper according to the expected environment and operational conditions. On the other hand, our approach can better exploit a goal-oriented approach when designing complex systems.

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

References

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

[2] 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*, pages 9–20, 7 2004.

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

[4] 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*, 2(4):357–401, 1999. Publisher: Kluwer Academic Publishers, Netherlands.

[5] M. J. Huber. Jam: A BDI-theoretic mobile agent architecture. In *Agents*, pages 236–243, 1999.

[6] 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*, pages 1–7, 2005. ACM Press, New York, NY, USA.

[7] 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)*, page 236, 2005.

[8] A. S. Rao. AgentSpeak(L): BDI agents speak out in a logical computable language. In R. van Hoe, editor, *Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, Eindhoven, The Netherlands, 1996.

[9] A. S. Rao and M. P. Georgeff. BDI agents: from theory to practice. In *Proceedings of the First International Conference on Multi—Agent Systems*, pages 312–319, San Francisco, CA, 1995. MIT Press.

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