A BDI Architecture for Goal Deliberation

Alexander Pokahr, Lars Braubach, Winfried Lamersdorf
Distributed Systems and Information Systems
Computer Science Department, University of Hamburg
{pokahr | braubach | lamersd}@informatik.uni-hamburg.de

ABSTRACT

One aspect of rational behavior is that agents can pursue multiple goals in parallel. Current BDI theory and systems do not provide a theoretical or architectural framework for deciding how goals interact and how an agent can decide which goals to pursue. Instead, they assume for simplicity reasons that agents always pursue consistent goal sets. By omitting this important aspect of rationality, the problem of goal deliberation is shifted from the architecture to the agent programming level and needs to be handled by the agent developer in an error-prone ad-hoc manner. This paper argues that goal deliberation mechanisms can hardly be built directly into the fixed BDI interpreter cycle, because goal deliberation typically needs to be done irregularly at any point in time. Therefore, an enhanced BDI interpreter architecture is proposed that is specifically designed for extensibility. This extensibility can be exploited for the integration of arbitrary goal deliberation strategies.

Categories and Subject Descriptors

I.2.11 [**ARTIFICIAL INTELLIGENCE**]: Distributed Artificial Intelligence—*Intelligent agents*

General Terms

Design

Keywords

BDI Agents, Goal Deliberation

1. INTRODUCTION

Goal-directedness is one important characteristic of rational agents, because it allows agents to exhibit pro-active behavior and it is argued that the BDI (belief-desire-intention) model [1] is well suited to describe this kind of agents. Typically, goal-directed agents should be capable of pursuing multiple goals simultaneously. As a consequence the agent's goals can interact positively or negatively with each other [6]. Positive interaction means that one goal contributes to the fulfillment of another one, whereas negative contribution indicates a conflict situation in which one goal hinders the other. Such contribution relationships between goals are commonly used in modeling agent applications, e.g. in the Tropos methodology [2] and in the requirements engineering technique KAOS [3]. Despite their usefulness, most implemented agent systems based on the BDI model do not support any mechanism for handling goal relationships at the architectural level. Hence, the cumbersome task of ensuring that the agent will never process any conflicting goals at the same time is left to the agent developer.

The main aspect of goal deliberation is "*How can an agent deliberate on its (possibly conflicting) goals to decide which ones shall be pursued?*". Considering this question from an architectural point of view it is of interest how a goal deliberation strategy can be integrated into a BDI infrastructure. Thereby, the agent infrastructure has the tasks to activate the strategy at certain points in time and to provide a clearly defined interface by specifying the possible operations for conflict resolution and exploiting positive goal interactions. These operations are constrained by the attitudes supported by the agent architecture. E.g. only when the architecture distinguishes between goals and desires the deliberation process can resort to both concepts.

The approach presented in this paper proposes a new flexible BDI-interpreter architecture. This architecture does not employ a fixed deliberation cycle, but instead resorts to an action-based model, in which the interpreter selects and executes BDI meta-actions like *execute plan step*. These meta-actions are directly derived from the well-known BDI abstract interpreter [5] and extended with goal-related actions (e.g. *create goal*) that support an explicit handling of goals.

2. A FLEXIBLE BDI INTERPRETER

The basic idea of the new architecture is to break up the original interpreter cycle as described in [5] into a small set of self-contained meta-actions, which are invoked as needed, rather than being executed in a fixed sequence. Instead of operating on global data structures, these actions are instantiated for individual attitudes and other elements such as events.

2.1 Abstract Interpreter Actions

Fig. 1 shows the identified actions (dark rectangles) and introduces abstract actions (light rectangles) as well as inheritance relationships (arrows) to group similar actions. The names of the abstract interpreter steps are given below the action names. The *Agent Init* action initializes an agent and is executed only once for each agent. One important responsibility of the architecture is to find and select applicable plans for an event or a goal. The corresponding actions are closely related and therefore grouped together by an abstract *Process Event Action*. The *Find Applicable Candidates* action determines the list of plans that are able to handle a given

AAMAS'05, July 25-29, 2005, Utrecht, Netherlands.

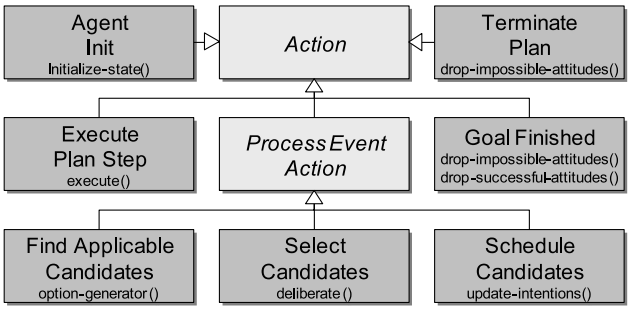

Figure 1: Identified meta-actions

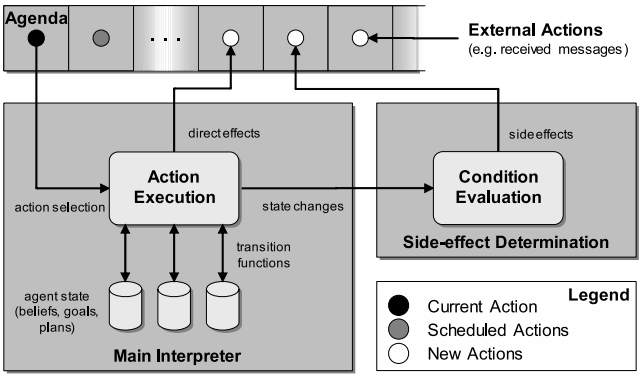

Figure 2: Interpreter architecture

event or goal. From this list a subset of plans to be executed has to be selected by the *Select Candidates* action. Selected plans then have to be scheduled for execution (*Schedule Candidates*). Any scheduled plan eventually has to be executed which is done in the *Execute Plan Step* action. According to the original interpreter only a single step is executed, allowing the agent to do other things before continuing with the plan. The *get-new-external-events* step of the original interpreter cycle is not represented as an action in itself, as new events are added on-the-fly. The final two steps of the interpreter are captured in the *Terminate Plan* and *Goal Finished* actions. Instead of handling failure and success in separate steps, the actions distinguish between mental attitudes (goals and plans).

This set of basic meta-actions is sufficient to rebuild the behavior of traditional BDI agents and can be easily extended with custom meta-actions for specialized agent architectures (e.g. supporting a certain kind of deliberation).

2.2 Interpreter Architecture

The set of meta-actions forms the basis of the new interpreter architecture. Abandoning the view that all actions are executed after each other in a fixed interpreter cycle, the question arises how can be decided which action to execute next, and also, when should new actions be instantiated.

The basic mode of operation of the proposed interpreter is depicted in Fig. 2 (left hand side). The interpreter is based on a data structure called *Agenda* where all actions to be processed are collected. The interpreter continuously selects the next entry from the agenda according to the interpreter's action selection strategy and executes it. The *action execution* may lead to the creation of new actions (*direct effects*), which are also inserted into the agenda. In addition, certain occurrences may render the execution of already scheduled actions obsolete, e.g. an execute plan step action for a meanwhile dropped goal should not be performed. Hence, a precondition can be assigned to an action to ensure that obsolete actions are not executed and instead removed from the agenda.

For certain actions such as drop impossible/successful attitudes in the original interpreter cycle, the operation itself has to decide if the conditions for dropping an attitude hold. In our action based model conditional actions separate the condition monitoring from the action part. Conditions are created already when the corresponding element is instantiated (e.g. a drop condition of a goal). Only when conditions trigger during the lifetime of the corresponding element, the associated action has to be executed. Therefore, another important part of the architecture is the *condition evaluation* (see Fig. 2, right hand side). This component checks system *state changes* against all currently existing conditions. For all conditions that are triggered by one or more of the state changes new agenda entries will be produced. These *"side-effects"* of the current action are subsequently added to the agenda. Additionally, external sources may also add entries to the agenda, such as messages that have been received from other agents and need to be processed.

The presented interpreter architecture builds the functional core of the extensively restructured Jadex BDI reasoning engine [4].

3. CONCLUSION

This paper tackles the question how goal deliberation strategies can be integrated into BDI agent systems. To facilitate flexibility and extensibility, a new BDI architecture is presented. The architecture is based on a flexible and backwards compatible interpreter executing meta-actions from a dynamic agenda. Besides the basic set of meta-actions derived from the traditional BDI interpreter, new meta-actions can be easily integrated to extend the architecture in various aspects. Hence, arbitrary goal deliberation strategies can be realized by providing new meta-actions specific for the strategy. The interpreter architecture automatically takes care of activating those actions at proper times.

Future work is on the one hand devoted to the further investigation of concrete deliberation strategies. We intend to experiment with a strategy we have implemented, and with other strategies (e.g. based on the work of Thangarajah et al). On the other hand we intend to exploit the flexible interpreter architecture in other directions, e.g. by investigating how other mental attitudes, such as obligations, could be integrated.

4. REFERENCES

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

[2] F. Giunchiglia, J. Mylopoulos, and A. Perini. The Tropos Software Development Methodology: Processes, Models and Diagrams. In *Proc. of 1st Int. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS'02)*, 2002.

[3] E. Letier and A. van Lamsweerde. Deriving operational software specifications from system goals. *SIGSOFT Softw. Eng. Notes*, 27(6):119–128, 2002.

[4] A. Pokahr, L. Braubach, and W. Lamersdorf. Jadex: A BDI reasoning engine. In R. Bordini, M. Dastani, A. Seghrouchni, and J. Dix, editors, *Multi-Agent Programming*. Kluwer, 2005.

[5] A. Rao and M. Georgeff. BDI Agents: from theory to practice. In *Proc. of the 1st Int. Conf. on MAS (ICMAS'95)*, 1995.

[6] J. Thangarajah, L. Padgham, and M. Winikoff. Detecting and Avoiding Interference Between Goals in Intelligent Agents. In *Proc. of the 18th Int. Joint Conf. on AI (IJCAI 2003)*, 2003.