REACTIVE (RE) PLANNING AGENTS IN A DYNAMIC ENVIRONMENT

Debdeep Banerjee, Jeffrey Tweedale
KES Centre, University of South Australia, Mawson Lakes, Australia, Debdeep.Banerjee@unisa.edu.au. Airborne Mission Systems Baranch, Defence Science and Technology Organisation Edinburgh, South Australia, Australia, Jeffrey.Tweedale@dsto.defence.gov.au .

Abstract: Intelligent agents are powerful tools for complex and dynamic problems. Belief Desire Intension (BDI) is one of the most popular agent architectures for reactive goal directed agents. Planning is intrinsic for intelligent behaviour. But planning from first principle is costly in terms of computation time and resources. BDI agents retain their reactive property by avoiding planning from real-time planning by using predefined plan library designed by agent designers. BDI agents look for a plan in the library to achieve their goals. If the agent could find a plan it fails to achieve the goal. It would be useful to have some real-time look ahead planning capability within BDI framework. In this paper we have proposed an architecture that includes (re) planning in BDI agents. The proposed architecture describes how to integrate a real-time planner with replanning capability in the current BDI architecture. Replanning capability is important for reactive behaviour.

Key words: BDI agent, AI planning

1. INTRODUCTION

Intelligent software agents are powerful tools in today's modern software systems. They have been deployed in complex and dynamic hostile environment and even in unknown environments. Research in intelligent agents is very active and progressive. According to Russell and Norvig [14]

"An agent can be anything that perceives its environment through sensors prior to acting upon the environment through actuators". Wooldridge defines an agent as "a computer system capable of autonomous action in a given environment in order to meet its design objectives" [1]. Jennings adds four main properties to these definitions which are: autonomy, reactivity, pro-activity and social ability [2]. Another important aspect to consider is its environment that an agent operates [14]. The agent's characteristics and capabilities also depend on the environment, because agent has to interact with its environment. Agent's environment has been divided between Static and Dynamic environment, fully Observable and partially Observable environment, Continuous and Discrete environment, Deterministic and Non-deterministic environment. There are mainly three common categories of architectures used to design intelligent agents: the Brooks subsumption architecture [17], Bratmans' Belief-Desires-Intension (BDI) architecture [4] and the layered model [18]. The BDI model was derived from the model of human practical reasoning system [4] based on rational agents that conduct actions that will help it achieve its goals. Practical reasoning is used to decide what to do (deliberation) and how to do it (means-end-analysis) [1].

Planning is intrinsic for any intelligent behaviour. Humans tend to plan most events they contribute to in the real world. The planning process may not always be visible, especially when those actions require routine skills, rule based tasks and procedures performed by subject matter experts. Humans' (re) use the rules, skills and knowledge stored in memory (plans that need to be embodied into agents) to achieve tasks they may have previously encountered [16]. We tend to plan for situations that are new, complex or critical. Planning is a costly process in the terms of time and computation. The motivation for Intelligent Agents is to personify human capabilities, so they can be used in place of humans and how they achieve the given goal by acting rationally in their environment. The main part of a rational act is that of practical reasoning. So the planning is the part of the practical reasoning as it describes a set of actions to achieve a goal [4].

This paper is organized in four parts. Section two describes the problems relating to planning in BDI architecture and the motivation of the proposed architecture. Next section describes the proposed architecture and section four contains the research methodology. Section five concludes the paper with the future directions.

2. MOTIVATION

BDI agents use a plan library or predefined set of plans, instead of planning from first principle [3]. When an agent commits to an intension, it

looks through its plan library for feasible plan, which is executed in order to achieve the goal. If the plan fails and a suitable alternative can't be identified then the agent fails to achieve the goal [3]. The main bottleneck of most BDI architectures is the plan library. Library agents are predominantly deterministic in nature, because all of its behaviour is hard coded into the library. The knowledge about how to achieve a specific task must be explicitly captured as plans in the library by the designer, prior to run-time. If the agent's environment is static, or partially dynamic and deterministic, then the above approach is efficient. In most cases the real environment is dynamic and non-deterministic. In this case it becomes extremely challenging for agent designer to write task specific plans for every possible situation. In this case a generic approach must be to guide the agent towards a possible solution.

Bratman used practical reasoning to construct his BDI architecture [4] when computer hardware had primitive capabilities with limited computational power. Modern computers enable designers to write larger and more complex agents. It also allows designers to relax some of the original resource constraint. Ideally an intelligent rational agent should be able do decide what to do and how to do it in a particular situation. The agent is designed to do a specific task (such as monitoring the communication network and fault diagnosis). Artificial Intelligent agents only need to decide how to achieve its goals by using knowledge about the environment and knowledge about its capabilities and measures. Within this process the agent should identify any sub-tasks to be achieved given the goal and suitable plans to achieve them. To succeed, autonomous agents must have a planning component capable of synthesizing its own course of actions from within the environment it resides.

There are agent architectures (such as RETSINA [19], PROPICE [20], CYPRESS [21], INTERRAP [22], TAIPE [23] etc.) that incorporate a planning component as part of the agent architecture. These systems implement different architecture to incorporate the planning module. Our proposed architecture extends the BDI agent architecture with online planning capabilities which will be handled within main BDI loop rather than accessed as an external component.

3. A FLEXIBLE PLANNING ARCHITECTURE FOR BDI AGENTS

To provide reactive behaviour to the BDI agents a new architecture has been proposed (Fig 2). This architecture is an extension of the BDI

architecture (Fig 1) as Wooldridge described in [3]. Two main modifications have been made in this architecture compare to the previous architecture.

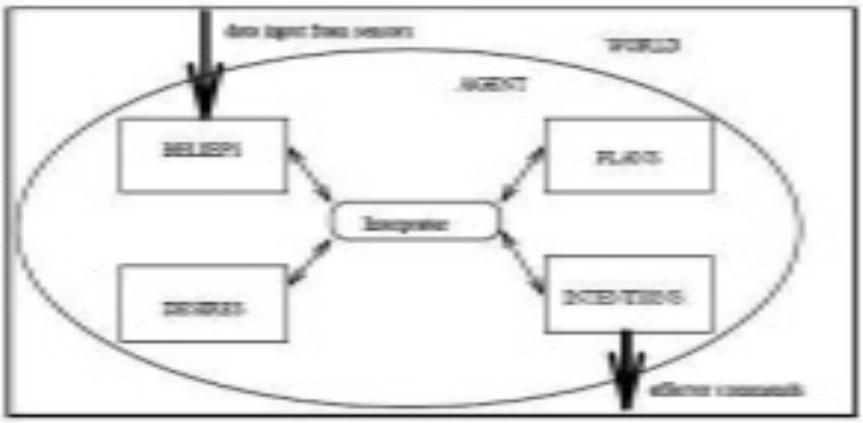

Figure 1. BDI Agent Architecture

The agent interpreter module has been extended by introducing a new State Change Monitor and a Sub Goal Deliberation module. The plan library of the previous architecture has been replaced by the Planning module. The purpose of these changes is to provide an agent means to react in a dynamic environment by reacting to the changes that occurs dynamically within that environment.

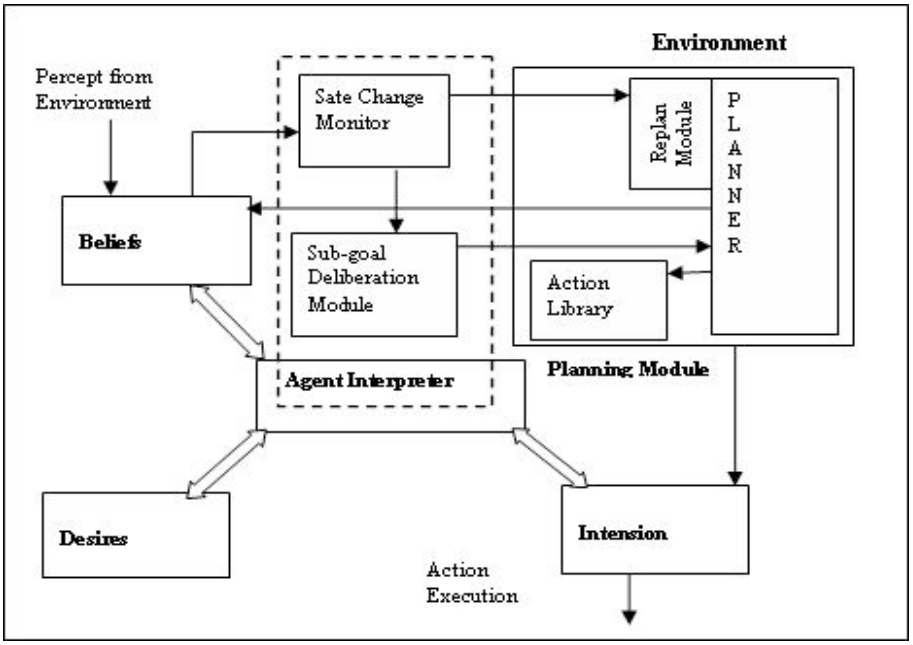

Figure 2. (Re) Planning Agent Architecture

3.1 The Planning module

To overcome the problems of the restricted plan library an online planning module has been introduced in the place of the plan library. The planning module consists of three sub-modules.

3.1.1 Action Library

It contains the actions that an agent can perform. Every action has preconditions and effects. These actions can be modeled as plans without any sub-plans. The practical implementation of the actions can be in different abstraction levels. This level of abstraction depends on the problem.

3.1.2 Planner

This can be any planner that will take the initial state, goal state and a set of action and synthesis a sequence of executable actions. Type of planner can depend on the problem domain.

3.1.3 Replanning Module

It is responsible for repairing or refining a failed plan. The output of the replanning module can be an abstract plan or a partial plan or even a total plan depends on the particular replanning strategy chosen.

3.2 The Extensions of the Agent Interpreter

To be reactive an agent should identify the changes in the environment that has an effect on its behavior. Then agent should decide what it should do to deal with the changes and how it can still achieve its goal. To provide this reactive deliberation capability Sate Change Monitor and Sub-goal Deliberation Module has been proposed as an extension of the agent interpreter.

3.2.1 Sate Change Monitor

It monitors the state of the world. It checks that if environment changes in such a way that it would make the some goal state true or assumptions of planning (conjunction of the preconditions of the actions) false. There can be two types of reactions from the state change monitor. Firstly it needs to identify when the goal is already been achieved or goal can not be achieved, then it stops the planning process and notify the Sub-goal Deliberation Module. Secondly when it identifies that some action preconditions of the plan become false it invokes the replanning module. Then the replanning module tries to repair the current plan. Using the State Change Monitor, we can separate the situation where we would need a new plan and where we need to repair the plan. Until the goal state is achieved or goal state become unachievable agent should try to achieve the goal.

3.2.2 Sub-goal Deliberation

It defines a goal sate for the agent by considering the current environment and agent's desires. It would take current world state and agent's desires as input and produce a desired goal state for the agent. It forwards the goal state to the planner. The Sub-goal Deliberation module can be implemented differently for different domains and can contain domain knowledge. Sub-goal Deliberation Module provides the goal state for planning. For goal state synthesis different approaches, such as decision theoretic approach, case-based approach, knowledge base, hierarchical task network approach etc, can be incorporated. The Sub-goal Deliberation module can be designed as

per the type of the environment and the problem. The level of the granularity of the goals is proportional to the dynamic nature of the problem.

This architecture will provide reactivity to the agent situated in a dynamic environment. Agent can handle the changes in the environment by monitoring the state of the world. This architecture is flexible and extensible. Different types of reactive domains (for example robot soccer, UAV, UT etc.) can be encoded by choosing different sub goal formulation methods in Sub-goal Deliberation module. Next section discusses the methodologies involved in developing the architecture discussed above.

4. METHODOLOGY

In the proposed agent architecture the extension of the agent interpreter would provide the method of Sub-goal Deliberation and the planning module would provide the means-end analysis. We divide the problem in three main parts. The first is designing the Sate Change Monitor module. The second is to provide a real-time planning module with replanning capability and third will be the modeling the Sub-goal Deliberation process.

For the first problem we need to design a module that will identify when a plan fails. There are two possibilities, the goal state has already been achieved by other agents or the goal state become unachievable and some precondition become false due to some changes in the environment. In the first case the State Change Monitor should send a notification to the Sub-goal Deliberation module for new goal deliberation and for the latter case it invokes the replanning sub module of the planning module. To implement a State Change Monitor we need to implement an execution monitoring module that checks if the plan is still consistent with the current world after execution of each action. To check consistency it will check if all the preconditions of the actions that are still to be executed at the next step are true and the goal is still achievable but not achieved yet [14].

For the planning module we assume the environment is fully observable. To incorporate the planning module we need to find a common representation of the actions between the chosen planner and the agent architecture. There are similarities between the BDI architecture and HTN (Hierarchical Task Network) based planning [5]. A wrapper can be created that maps the BDI agent syntax to the HTN based planner syntax similar to [6]. For replanning capabilities we can implement a replanning module on top of the planner. There are different options exist for replanning. First option would be introducing replanning algorithm [8, 9] which can start replanning by backtracking from the point where the plan fails and choose

alternative path. Replanning can also be done by plan refinement technique where in the case of a failed plan refinement technique replaces the failed actions with the alternate actions [10, 11]. In the dynamic environment the environment changes very fast. An agent situated in this dynamic environment needs to react to these changes. For this highly reactive behaviour agent may not need to synthesis a full plan for achieving goal. This reactivity can be achieved by incorporating anytime algorithm based planner [7]. In anytime based planner a planner can be interrupted at anytime and planner always have some executable plan as the result [7, 8]. The main problem in this kind of planner is to guarantee the quality of the resultant plan. On the other hand genetic algorithm [12, 13] can also be implemented so at any point of time agent can have an executable plan.

The Sub-goal Deliberation module can be compared to the plan library of the BDI agent architecture. It can contain the domain specific knowledge in the form of predefined task decomposition. Only difference would be instead of producing an executable plan it will produce abstract level tasks as sub goals. We can incorporate different strategies, such as decision theoretic approach, case-based approach, knowledge-based approach, for Sub-goal Deliberation. Sub-goal Deliberation process can be modeled as planning problem that will generate abstract sequence of tasks and the planning module can be seen as action scheduling problem for instantiating those tasks.

5. CONCLUSION

The current BDI model's main bottleneck is the plan library. If the agent fails to find a plan in its plan library it fails to achieve the goal. This is not desirable in most real world situations. The agent must adapt to the current situation. Since most of the real world environment is complex and highly dynamic it is nearly impossible for an agent designer to write predefined plan for every possible situations. The proposed architecture introduced online planning with replanning capability in BDI agent architecture. This architecture can use the domain knowledge for Sub-goal Deliberation and provide flexibility for different types of dynamic domains. This architecture can also be extent in the cases where the environment is not fully observable and changes frequently in random manner.

The implementation phase has four main steps. The first step is to find a common representation of the planning problem between the planner and the agent architecture. We will use JACK as our BDI implementation. JACK [15] is a BDI based commercial strength multi-agent based software development framework based on JAVA. It is developed by the Agent

Oriented Software. JACK provides a high performance, lightweight implementation of BDI architecture. It is an agent oriented programming extension of JAVA. The second step would be interfacing a external planner with the JACK agent or incorporating an planning algorithm within JACK. The next step would be implementing a State Change Monitor in context of JACK system. Last step would be to extend the planner with some replanning or anytime algorithm for reactivity. Different replanning algorithm can implemented and compared based on the performance.

REFERENCE

1. M. Wooldridge: *Reasoning about Rational Agents*, The MIT Press, London (2000)
2. M. Wooldridge, N.R. Jennings: *Intelligent agents- theory and practice*, Knowledge Engineering Review, 10 (2), (1995)
3. Wooldridge, M: *Practical Reasoning with Procedural Knowledge- A Logic of BDI Agents with Know-How,* in Proceedings of the International Conference on Formal and Applied Practical Reasoning, Springer-Verlag, Berlin (1996)
4. Bratman ME: *Intentions, Plans and Practical Reason*, Harvard University Press: Cambridge, MA (1987)
5. Lavindra de Silva and Lin Padgham: *A Comparison of BDI Based Real-Time Reasoning and HTN Based Planning*. In Proceedings of the 17th Australian Joint Conference on Artificial Intelligence, Cairns, Australia, (Dec 2004)
6. Lavindra de Silva, Lin Padgham: *Planning on Demand in BDI Systems*. International Conference on Automated Planning and Scheduling, Monterey, California, (June 2005)
7. N. Hawes: *Anytime planning for agent behaviou*r, In Proceedings of the 12th Workshop of PLANSIG, (2001)157-166
8. Hawes. N: *An anytime planning agent for computer game worlds*. In Workshop on Agents in Computer Games at The 3rd International Conference on Computers and Games (CG'02), Edmonton, Canada, (2002) 1-14
9. G. Boella, R. Damiano: *A replanning algorithm for a reactive agent architecture*. In D. Scott, editor, Artificial Intelligence: Methodology, Systems, and Applications*, LNCS 2443*, Springer Verlag, (2002) 183-192
10. Roman van der Krogt and Mathijs de Weerdt: *Plan Repair using a Plan Library*, BNAIC, (2005) 284-259
11. Roman van der Krogt, Mathijs de Weerdt: *Plan Repair as an Extension of Planning*. ICAPS,(2005) 161-170
12. C. H. Westerberg, J. Levine: *GenPlan- Combining genetic programming and planning*. In Proc. of the 19th Workshop of the UK Planning and Scheduling Special Interest Group (PLANSIG), (2000)
13. L. Spector: *Genetic programming and AI planning systems*, Proceedings of the twelfth national conference on Artificial intelligence, Seattle, Washington, United States, (1994)
14. S. J. Russell, P. Norvig: *Artificial Intelligence- A Modern Approach*. Prentice Hall, 2nd edition, (2003)
15. A. Hodgson, N. Howden, R. Rönnquist and A. Lucas: Jack intelligent agents -- summary of an agent infrastructure. In 5th International Conference on Autonomous Agents (2001)
16. Rasmussen, J: Information processing and human machine interaction: An approach to cognitive engineering, New York, North Holland (1986).

17. R. A. Brooks: *How to build complete creatures rather than isolated cognitive simulators*. In K. VanLehn (ed.), Architectures for Intelligence, Lawrence Erlbaum Assosiates, Hillsdale, NJ (1991) 225-239
18. J. P. Mller and M. Pischel: *The Agent Architecture InteRRaP: Concept and Application*. Technical Report RR-93-26, DFKI Saarbrucken, (1993)
19. M. Paolucci, D. Kalp, A. Pannu, O. Shehory and K. Sycara: *.A planning component for RETSINA agents*"; Lecture Notes in Artitcial Intelligence, Intelligent Agents VI, Springer (2000).
20. O. Despouys and F. F. IngrandF: *Propice-Plan: Toward a Unified Framework for Planning and Execution*. In European Conference on Planning (ECP), 278-293, 1999[1]
21. D. Wilkins, K. Myers, J. Lowrance, and L. Wesley: *Planning and reacting in uncertain and dynamic environments*, Journal of Experimental and Theoretical Artificial Intelligence (7) (1995) 972-978.
22. K. Fischer, J.P. Muller and M. Pisschel. *Unifying control in a layered agent architecture*, Agent Theory, Architecture and Language Workshop, Montreal (1995)
23. E. H. Durfee, M. Huber, M. Kurnow and J. Lee, *TAIPE: Tactical Assistants for Interaction Planning and Execution*, First International Conference on Autonomous Agents (Agents'97) (1997).