International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 2 Issue 2, February- 2013

BDI: Applications and Architectures

Dr. Smitha Rao M.S, Jyothsna.A.N
Department of Master of Computer Applications
Reva Institute of Technology and Management
Bangalore, India

Abstract

Today Agent Technology is used in designing complex systems. It is a difficult task to be able to build agents in a believable way, as their behaviour is led by many conflicting needs and goals. BDI (Belief-Desire-Intention) paradigm is a widely used mechanism to formalize the internal architecture of complex agents. Using BDI we can design realistic agents which are also expressive. Though this paradigm is easy to understand and relate to, it is difficult to build as they are complex in nature. Agents developed using BDI paradigm are used in several applications such as Air Traffic Management, e-Health Applications, Combat Air Mission Reasoning and Control, Automation of customer service application etc. Our paper focuses on the various aspects of BDI paradigm by providing the case study of an intelligent web spider based on this architecture.

1. Introduction

Since couple of years Agent technology has been the buzz word in the minds of many researchers. People involved in agent research have varied definitions for the word Agent. The main characteristic of an agent is that it acts on behalf of others. An agent can be broadly defined as a logical unit of an application or a self-contained program which is capable of controlling its own decision making and acting, based on its perception of its environment, in pursuit of one or more goals. An agent based system works in an open unpredictable environment and is dynamic and flexible. Agent-oriented methodologies frequently make use of terms such as goals and tasks but do so in an inconsistent manner. Two models that have been widely used in the implementation of autonomous agents are, the Belief Desire Intention (BDI) model and the Markov Decision Processes (MDPs) model. Markov Decision Process is based on Markov's Property. A stochastic process has the Markov property if the conditional probability distribution of future states of the process depends only upon the present state, not on the sequence of events that preceded it. On the other hand, BDI paradigm is based on folk psychology. Also known as commonsense psychology, folk psychology is the natural capacity to explain and predict the behaviour and mental states of other people. According to this psychology the best way to describe a complex system is through providing references to familiar terms and items. This implies that the core concepts of an agent framework map easily to the language people use to describe their reasoning and actions in everyday conversation.

BDI is a software model developed for programming intelligent agents. BDI software model implements the principal aspects of Michael Bratman's theory of human practical reasoning (also referred to as Belief- Desire- Intention or BDI)[6]. It is a way of explaining future-directed intention, and has been applied as a way of limiting the time spent deliberating on what to do by eliminating choices inconsistent with current intentions. BDI agents are situated in a changing environment, they receive continuous perceptual input, and take actions to affect their environment, all based on their internal state. From its inception in 1980's, BDI has not seen considerable growth till recent times. Today the usage of multi-agent goal oriented systems show a need for a BDI based framework. Beliefs, desires, and intentions are the three primary mental attitudes and they capture the informational, motivational, and decision components of an agent, respectively. Beliefs represent the agent's current knowledge about the world, including information about the current state of the environment inferred from perception devices and messages from other agents, as well as internal information. Desires

represent a state which the agent is trying to achieve. Intentions are the chosen means to achieve the agent's desires, and are generally implemented as plans and post-conditions. Thus simply putting it, Beliefs consist of what the agent believes to be true about the current state of the world, desires consist of the agent's goals, and intentions consist of the agent's current plan for achieving those goals. BDI software model is closely associated with intelligent agents, but does not carter to all the facets of intelligent agents like for example inter agent communication. BDI paradigm has been applied in our paper to discuss the architecture of an intelligent Web Spider that scans the web searching for pirated copies of content like video, image etc.

Organization of the paper is as follows. BDI applications are presented in Section 2. Architectures and Languages use in BDI paradigm are presented in Section 3. The architecture or an intelligent web spider based on the BDI paradigm is discussed in Section 4. Finally, conclusions are summarized in Section 5.

2. BDI Applications

BDI agents have been used with considerable success to model humans and create human-like characters in simulated environments. One of the reasons for growing success of agent-based technology is that it has been shown to be quite useful in the development of various types of applications, including air-traffic control, autonomous spacecraft control, health care services, industrial control systems etc. One of the popular frameworks based on BDI paradigm is PRS (Procedural Reasoning System). PRS has been deployed in many major industrial applications such as fault diagnosis on the space shuttle [1], air traffic management, business process control [2] etc. Some of the successful implementations of PRS are Oasis and SWARMM. Oasis (Optimal Aircraft Sequencing using Intelligent Scheduling) was tested successfully at Sydney Airport in 1995. It was a system for air traffic management that could handle the flow of aircrafts arriving at an airport. The system dealt with issues like aircraft scheduling, comparing actual progress with established sequences of aircraft, estimating delays, and notifying controllers of was to correct deviations. The prototype implementation of Oasis comprised of several different kinds of agents, like aircraft agents, coordinator agents etc., each of these agents were based around PRS.

SWARMM(Smart Whole AiR Mission Model) has been used as the basis of an agent-based simulation system developed for Australia's Defence Science and Technology Organization, to simulate air mission dynamics and pilot reasoning. SWARMM was built using dMARS (distributed Multi Agent Reasoning System) which is one of the implementations of PRS. DSTO (Defence Science and Technology Organization) of Australian Department of Defence used SWARMM for defence studies. DSTO later replaced dMARS with JACK intelligent agents.

Norwegian-based Statoil, which is one of the world's largest suppliers of crude oil and natural gas, has developed software to support oil trading and operations management, using JACK. Intelligent agents are being applied to solve optimization, planning and process control issues in Statoil's trading and operation areas.

More recent work has been the application of dMARS to represent different roles in an organization in more general business software for running call centres and internet services.

3. Architecture and Languages

Since the mid 1980s, many control architectures for practical reasoning agents have been proposed [4]. Programming a BDI-based agent amounts to specifying its initial state in terms of beliefs(information), goals(objectives), and plans(means). In programming terminology, the beliefs, goals, and plans can be considered as data structures specifying the state of the agent program. The execution of a BDI-based agent, which is supposed to modify the state of the agent program, is based on a cyclic process called deliberation cycle (sense-reason-act cycle). Each iteration of this process starts with sensing the environment (i.e., receive events and messages), reasoning about its state (i.e., update the state with received events and messages, and generate plans to either achieve goals or to react to events), and performing actions (i.e., perform actions of the generated plans). The various agent architectures highlighted in this paper are PRS (Procedural Reasoning System) and variants such as AgentSpeak(L), JASON, JAM, dMARS, and JACK Intelligent Agents.

3.1. PRS

The earliest implementation of BDI paradigm was Procedural Reasoning System (PRS). It was developed by George and Lansky [3].This architecture has progressed from an experimental LISP version to a fully fledged C++ implementation known as the distributed Multi-Agent Reasoning System (dMARS), which has been applied in perhaps the most significant multi-agent applications to date [5]. PRS is a framework for constructing real-time reasoning systems

that can perform complex tasks in dynamic environments. PRS was developed for embedded application in dynamic and real-time environments. Various members of the PRS family of BDI agent systems differ from one another in terms of their implementation and features but they are all based on similar interpretation of BDI. The internal structure of PRS is composed of database (beliefs), goal intention structure, KA (Knowledge Area), library plans, interpreter (reasoned) etc. An application of PRS was also used as monitoring and fault detection system for the reaction control system on the NASA space shuttle.

3.2. AgentSpeak (L)

In the area of agent-oriented programming languages, AgentSpeak (L) has been one of the most influential abstract languages based on the BDI paradigm. AgentSpeak (L) programming language was introduced in [7].It has a neat notation and is a computationally efficient extension of logic programming to BDI agents [8, 9]. An AgentSpeak agent is defined by a set of beliefs giving the initial state of the agent's belief base, which is a set of ground atomic formula, and a set of plans which form its plan library. A plan also has a body, which is a sequence of basic actions or goals that the agent has to achieve when the plan is triggered. AgentSpeak distinguishes two types of goals: achievement goals and test goals. An achievement goal states that the agent wants to achieve a state of the world where the associated atomic formula is true. A test goal states that the agent wants to test whether the associated atomic formula is one of its beliefs.

An AgentSpeak agent is a reactive planning system. The events it reacts to are related either to changes in beliefs due to perception of the environment, or to changes in the agent's goals that originate from the execution of plans triggered by prior events. In its original definition [7], AgentSpeak was just an abstract programming language. It was used for the formalization of ideas behind BDI architecture using modal logics.

3.3. JASON

Jason is the first fully-fledged interpreter for a much improved version of AgentSpeak. Jason initially stood for `Java-based AgentSpeak interpreter used with SACI for multi-agent distribution over the internet. A Jason based multi-agent system can be distributed over a network effortlessly. The main difference between the language interpreted by Jason and the original AgentSpeak (L) language is that, wherever an atomic formula was allowed in the original language, in Jason, a literal is used instead. The implementations that are currently available for agent communication in Jason are largely inspired by KQML. Jason is distributed with an Integrated Development Environment (IDE) which provides a GUI for editing a MAS configuration file as well as AgentSpeak code for the individual agents. Jason is implemented in Java (thus multi-platform) and is available Open Source, distributed under GNU LGPL. While using Jason there is no issue with portability, but very little consideration has been given so far to standards compliance and interoperability.

3.4. JAM

JAM is a hybrid intelligent agent architecture that is based upon the theories and ideas of the PRS, Structured Circuit Semantics (SCS) [11], and Act plan Interlingua [12, 10]. JAM provides rich and extensive plan, procedural representations, utility-based reasoning over multiple simultaneous goals, that are both goal and event driven. Each JAM agent is composed of five primary components: a world model, a plan library, an interpreter, an intention structure, and an observer. A JAM agent's behaviour is motivated by specifying top level goals. Goals can be given to the agent in a text form. JAM provides many programming actions and constructs. JAM agents facilitate building applications requiring mobility through the usage of checkpoint capabilities. There are a number of predefined primitive actions included with JAM agent distribution; including those providing debugging support and agent mobility.

3.5. dMARS

Distributed Multi-Agent Reasoning System (dMARS) dMARS is a C++ implementation of PRS. In dMARS, agents use plans to implement the BDI model. Each agent has a plan library, which is a set of plans, specifying courses of action that may be undertaken by an agent in order to achieve its intentions. An agent's plan library represents its procedural knowledge, i.e. the knowledge of how to bring about states of affairs. dMARS agents monitor both the world and their own internal state. Any

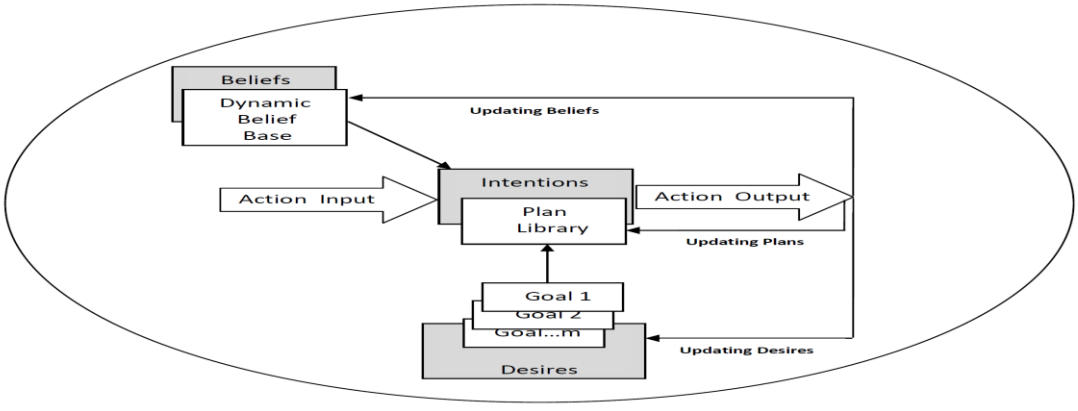

Figure 1. Architecture of our BDI based intelligent Web Spider

events that are perceived are placed on an event queue.

3.6. JACK

JACK is based on BDI paradigm and was built for simulations, in particular defence simulations. It is based on Java with a few syntactic extensions, and when compiled compiles to Java code [5]. JACK Intelligent Agents were initially developed in 1997 by ex-members of the Australian Artificial Intelligence Institute. JACK Intelligent Agents is a commercial multi-agent platform that has been under active research and development. JACK platform has been extended number of times since its inception. Most of the extensions, such as JACK Teams and CoJACK were developed in collaboration with AOS. JACK applications consist of a collection of autonomous agents that take input from the environment and communicate with other agents. This provides system builders with a very powerful form of encapsulation. Each agent is defined in terms of its goals, knowledge and social capability, and is then left to perform its function autonomously within the environment it is embedded in. As it is entirely written in java, JACK is highly portable and runs on anything from PDAs to high-end, multi-CPU servers. Its Java foundation means that JACK can run with multiple threads across multiple CPUs, has platform-independent GUIs, and is easily integrated with third-party libraries.

4. Case Scenario of Intelligent Web Spider

Web Spiders or web crawlers are automated computer programs that methodically crawl through the World Wide Web gathering required information in an orderly manner. Many search engines, use crawling or spidering as a means of providing up-to-date or current data. Spiders create a copy of all the visited pages for later processing by a search engine that will index the downloaded pages to provide fast searches. Crawlers can also be used for automating maintenance tasks on Web sites, to gather specific types of information from Web pages etc. We have considered an intelligent web spider based on BDI concept that scans the web, searching for pirated copies of watermarked images. Watermarking is done in an invisible fashion which is used to identify the pirated copies and solve copyright issues. The architecture of our BDI based web spider is shown in Figure 1. Web Spider draws its beliefs from a dynamic belief base. The desires of the spider agent are categorized as various goals. These goals could be independent or could be part of larger goals. Intentions of the Web Spider are plans which are part of the plan library.

Beliefs:

This is the informative component of the system like the list of the valid users, license details, watermark details with required id information, characteristics of the environment like IP address of the hosts visited by the spider etc. The belief base is updated appropriately after each plan is executed. For example if an action of detecting watermark in a file in one of sites results in a failure then the belief base would be updated with the IP address of the host as a potential black listed or rouge site containing pirated copies. Such a dynamic feature provided to the belief base would enhance the action plan of the spider in its consecutive scan of the host IPs.

Desires:

These are various goals meant to be achieved by the spider. These could be independent goals like scan a particular host for watermarked contents or could include sub goals like extract watermark, match the

extracted watermark with the data from belief base etc. Further goals would include creating an itinerary for the spider to travel, scanning the host to gather information of the visited environment etc. Each goal is designed keeping in view the desires of the web spider. Goals that make up the desires of the agent should not be conflicting. Conflicting goals would result in ambiguity regarding the choice of proper plan of action.

Intentions:

Each goal associated with a desire would require a course of action to be taken to achieve the target. Intentions represent the currently chosen course of action (the output of the most recent call to selection plan of action). Plans thus formed exist in the plan library. Examples for plans for a web spider would include; implementation of watermark detection algorithm, access to the belief base to verify the validity of the watermark, updating the belief base, plan library as well as the goals to create a more effective open system.

5. Conclusions

Agents are an emerging technology that has the potential to take over traditional methods for designing, and implementing complex software systems. The Belief Desire-Intention (BDI) agent paradigm has proven to be one of the major approaches to building intelligent agent systems in the industry. Typical BDI agent-oriented programming languages rely on user-provided plan libraries to achieve goals based on beliefs. BDI based systems are extremely flexible and responsive to the environment, and as a result are well suited for complex applications with real-time reasoning and control requirement. In a hybrid network environment BDI provides a better framework to develop intelligent automated agents.

6. References

[1] F. Ingrand, M. Georgeff, and A. Rao, ”An architecture for real-time reasoning and system control”, IEEE Expert, vol. 7, no. 6, pp. 34–44, 1992.
[2] M. P.Georgeff and A. S. Rao, “A profile of the Australian AI Institute”, IEEE Expert, vol. 11, no. 6, pp. 89–92, 1996.
[3] M. P. Georgeff and A. L. Lansky, ‘‘Reactive reasoning and planning,’’ in Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87), Seattle, WA, 1987, pp. 677–682.
[4] M. Wooldridge and N. R. Jennings, ‘‘Intelligent agents: Theory and practice,’’ Knowl. Eng. Rev., vol. 10, no. 2, pp. 115–152, 1995.
[5] P.Busetta, R. R¨onnquist, A. Hodgson, and A. Lucas. Light-Weight Intelligent Software Agents in Simulation. Technical Report 3, Agent Oriented Software, October, 1999.
[6] M.E. Bratman, D. J. Israel, and M. E. Pollack, “Plans and resource-bounded practical reasoning,” Computational Intelligence, vol. 4, pp. 349–355, 1988.
[7] A. S. Rao. AgentSpeak(L): BDI agents speak out in a logical omputable language. In W. Van de Velde and J. Perram, editors, Proceedings of the Seventh Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW’96), 22–25January, Eindhoven, The Netherlands, number 1038 in Lecture Notes in Artificial Intelligence, pages 42–55, London, 1996. Springer-Verlag.
[8] M. Wooldridge. Reasoning about Rational Agents. The MIT Press, Cambridge, MA, 2000.
[9] M.P.Singh, A. S. Rao, and M. P. Georgeff. Formal methods in DAI:Logicbased representation and reasoning. In G. Weiß, editor,Multiagent Systems—A Modern Approach to Distributed Artificial Intelligence, chapter 8, pages 331–376. MIT Press, Cambridge, MA, 1999.
[10] K.L.Myers and D. E. Wilkins. The Act Formalism, Version 2.2. SRI International Artificial Intelligence Center Technical Report, Menlo Park, CA, 1997.
[11] J. Lee, M. J. Huber, E. H. Durfee, and P. G. Kenny. UM-PRS: An Implementation of the Procedural Reasoning System for Multirobot Applications. In Conference on Intelligent Robotics in Field, Factory, Service, and Space (CIRFFSS’94), 842-849, Houston, Texas, 1994.
[12] D. E. Wilkins and K. L. Myers. A Common Knowledge Representation for Plan Generation and Reactive Execution. In Journal of Logic and Computation, vol. 5, number 6, 731-761, 1995.