Arquitetura BDI: Uma visão geral

Cláudia F. Bratficher Dário

Faculdade de Engenharia Elétrica e da Computação

claudia@ft.unicamp.br

Resumo – Os Sistemas Multi-Agentes representam uma área de grandes pesquisas e desenvolvimentos dentro da Engenharia de Software Orientada a Agentes. Os agentes de software com a capacidade de raciocínio passaram a ser utilizados em uma variedade de aplicações. A arquitetura BDI é uma arquitetura de agentes deliberativos baseados em estados mentais *belief-desire-intention*. Diversas plataformas para a arquitetura BDI foram propostas. Este trabalho apresenta uma visão geral da Arquitetura BDI e algumas plataformas de implementação.

PALAVRAS-CHAVE: AGENTE, ARQUITETURA BDI, PRS

I. INTRODUÇÃO

O desenvolvimento de sistemas de software utilizando agentes, denominados de Sistemas Multi-Agentes (SMA) tem crescido como um paradigma de computação, abrangendo uma grande variedade de aplicações, variando desde pequenos sistemas até sistemas mais complexos e de missão crítica. Apesar de atualmente ser um termo bastante utilizado, ainda não existe um conceito unânime de agente. A arquitetura de um agente especifica como este pode ser decomposto em um conjunto de módulos e suas interações descrevem como que os dados recebidos do ambiente e o estado interno do agente determinam suas ações [Wooldridge 1995]. Dessa forma, a arquitetura de um agente é a descrição dos processos internos que regem a interação do mesmo com o seu ambiente. Wooldridge classifica a arquitetura de agentes em três tipos: arquitetura reativa, arquitetura cognitiva ou deliberativa e arquitetura híbrida. Na arquitetura reativa não existe um modelo simbólico do mundo e não é utilizado nenhum raciocínio simbólico complexo. O modelo de funcionamento de um agente reativo é formado por estímulo-resposta. A tomada de decisão do agente reativo é implementada através do mapeamento direto de uma situação para ação. Na arquitetura deliberativa ou cognitiva existe um modelo simbólico do mundo representado explicitamente e as decisões relativas às ações a serem tomadas são feitas através de raciocínio lógico, baseado em reconhecimento de padrões e manipulação simbólica. A arquitetura híbrida, também conhecida como Arquitetura em Camadas, combina componentes das arquiteturas reativa e deliberativa. Este trabalho aborda a arquitetura cognitiva chamada BDI (*Belief, Desire and Intention*), incluindo algumas de suas implementações.

II. MODELO BDI

O modelo BDI foi originalmente proposto por Bratman [Bratman 1987] como uma teoria filosófica sobre o raciocínio prático, onde o comportamento humano é modelado com as seguintes atitudes: crenças, desejos e intenções. De acordo com o modelo BDI, as ações são derivadas do processo de raciocínio prático, constituído de dois passos. No primeiro passo, chamado de deliberação, é feita uma seleção de um conjunto de desejos que devem ser alcançados. No segundo passo é definido como esses desejos podem ser atingidos. O raciocínio prático pondera as considerações conflitantes a favor e contra alternativas competitivas, onde as considerações relevantes são determinadas pelo o que o agente acredita, importa e valoriza [Wooldridge 2000].

A seguir são detalhadas as três atitudes mentais que compõem o modelo BDI:

1. Crenças (*Beliefs*): representam as características do ambiente que são atualizadas após a percepção de cada ação. Representam o componente informativo.

2. Desejos (*Desires*): contém a informação sobre os objetivos a ser atingido. Representam o estado emocional do sistema.
3. Intenções (Intentions): contém o atual plano de ação escolhido. Representam o componente deliberativo do sistema.

III. ARQUITETURA BDI

Rao e Geogeff [Rao 1995] adotaram o modelo BDI para agentes de software e desenvolveram a arquitetura BDI, apresentando uma teoria formal e um interpretador BDI. A seguir são apresentadas as atitudes mentais adotadas na arquitetura BDI:

A. Crenças (Beliefs)

As crenças representam a visão fundamental do agente em relação ao seu ambiente. Uma crença é um estado mental intencional fundamental para as interações dos agentes com noção idêntica a de conhecimento. Um agente pode ter crenças sobre o mundo habitado por ele, sobre outros agentes, sobre interações com outros agentes e crenças sobre as suas próprias crenças. As crenças podem ser até contraditórias. Uma vez que o mundo é dinâmico, os eventos passados precisam ser lembrados e são representados pelas crenças. Podem ser vista como o componente informativo do sistema.

B. Desejos (Desires)

Os desejos representam os estados desejáveis que o sistema poderia apresentar, ou o estado motivacional do agente. O agente escolhe os desejos que são possíveis de acordo om algum critério, incentivando a realizar as tarefas para as quais ele foi projetado. Dessa forma, os desejos motivam o agente a agir para realizar as suas metas. Para que um desejo se realize é necessário o conhecimento inerente (a crença) e também o contexto favorável à sua realização. Um desejo pode estar em conflito com as crenças do agente ou com outro desejo.

C. Intenções (Intentions)

As intenções representam o atual plano de ação escolhido, correspondendo aos estados do mundo que o agente quer efetivamente provocar, ou seja, existe um comprometimento em realizá-las. Podem ser consideradas um subconjunto dos desejos, mas devem ser consistentes. Se um agente decide seguir uma meta específica, então esta meta torna-se uma intenção. As intenções determinam o processo de raciocínio prático, pois determinam as ações a serem realizadas. Uma vez escolhida uma intenção, haverá um direcionamento do raciocínio prático do futuro, com o objetivo de realizá-la. Na teoria de raciocínio prático, o processo de decidir que metas devem ser alcançadas é chamado deliberação, e o que é feito para alcançar estas metas é chamado de processo meio-fim. O agente reconsidera suas intenções de tempos em tempos para confirmar o alinhamento de suas intenções com seus desejos. É necessário que essa atividade de reconsideração seja realizada de maneira a não atrapalhar o funcionamento do agente, pois, se por um lado ela é necessária para que o agente não trabalhe por uma intenção desnecessária, por outro ela pode evitar que o agente realize suas intenções [Nunes 2007]. As intenções são formadas a partir de um processo de deliberação e a partir do refinamento de outras intenções, mas também podem conter as intenções iniciais inseridas pelo usuário.

Em Weiss [Weiss 1999] são definidos os seguintes componentes importantes de uma arquitetura BDI:

- Um conjunto de crenças representando as informações que o agente tem sobre seu ambiente atual;
- Uma função de revisão de crenças. Um novo conjunto de crenças é formado a partir das entradas dos dados coletados do meio e das crenças atuais do agente;
- Uma função geradora de opções, que determina as opções disponíveis para o agente, ou seja, seus desejos, tendo como base suas crenças atuais sobre seu ambiente e suas intenções atuais;
- Um conjunto de opções representando os desejos atuais do agente;
- Uma função filtro de deliberação que determina as intenções dos agentes, tendo como base suas crenças, desejos e intenções atuais;
- Um conjunto de intenções atuais, representando o foco atual do agente;

- Uma função de seleção de ação, responsável por determinar a ação a ser executada pelo agente baseada no conjunto de intenções atuais.

A figura 1 representa os componentes da arquitetura DBI [Nunes 2007].

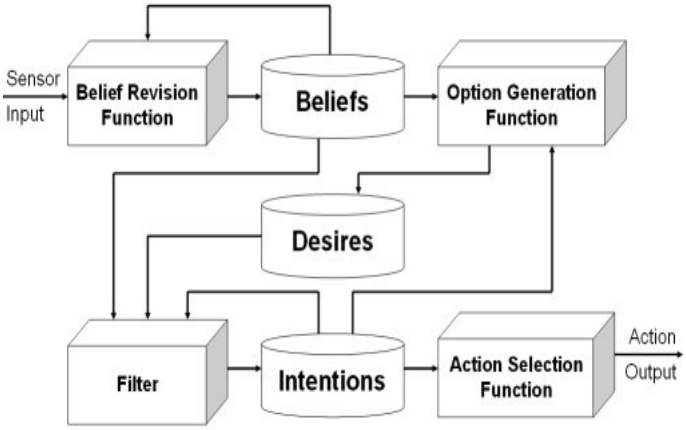

Figura 1: Diagrama da Arquitetura BDI

II. Implementações da arquitetura BDI

Existem uma série de implementações para a Arquitetura BDI. Várias delas foram desenvolvidas somente em ambientes artificiais e algumas foram aplicadas em problemas reais.

A.Procedural Reasoning System (PRS)

Procedural Reasoning System é um *framework* para a construção de sistemas de raciocínio em tempo real que podem realizar tarefas complexas em ambientes dinâmicos. É o primeiro sistema baseado no intepretador BDI, desenvolvido por Rao e Georgeff [Rao 1995]. Originalmente foi escrita em 1987, em linguagem LISP. A arquitetura do PRS inclue os seguintes componentes [Nunes 2007]:

- Banco de dados para as crenças do mundo, representado usando predicado de primeira ordem;
- Objetivos a serem realizados pelo sistema;
- Planos ou *Knowledge areas* (KAs) que definem a sequência de ações de baixo nível e testes para atingir os objetivos. A versão original do PRS utilizava KAs e versões mais recentes utilizam a representação de ações através de Atos;
- Intenções que incluem quais KAs foram selecionados para execução;
- Interpretador ou mecanismo de inferência que gerencia o sistema, escolhendo os planos mais apropriados.

O funcionamento do *PRS* está fundamentado na especificação das ações que serão usada para realizar os desejos. A base de dados armazena as crenças do sistema e as informações sobre o domínio da aplicação. O conhecimento contido na base de dados é expresso em lógica de primeira ordem. Os objetivos correspondem aos desejos atuais do sistema. É possível existir objetivo intrínseco, que não está relacionado a intenções existentes e objetivo operacional, que é um sub-objetivo de alguma intenção existente [Moser Fagundes]. A aplicação inicial do PRS foi um sistema de detecção de monitoramento e falhas para o sistema de controle de reação (*RCS*) no ônibus espacial da NASA. A figura a seguir ilustra o PRS.

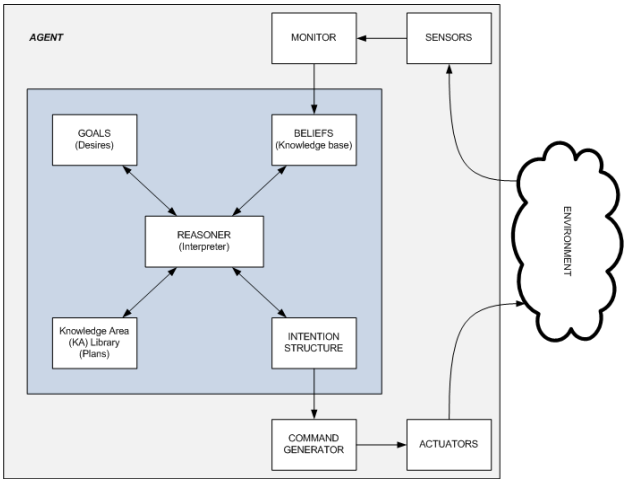

Figura 2: Procedural Reasoning System [PRS]

B.dMARS

O *dMARS* (*Distributed Multi-Agent Reasoning System*) foi desenvolvido em C++, é uma extensão do *Procedural*

Reasoning System (PRS) [d´Inverno 1997]. É um framework para a construção de sistemas de domínios dinâmicos onde há conhecimento incerto e complexo.

Os agentes *dMARS* monitoram o mundo, seus estados internos próprios e quaisquer eventos que são identificados e colocados em uma fila de eventos. O interpretador gerencia as operações dos agentes, executando o seguinte ciclo:

- observa o mundo e o estado interno do agente e atualiza a fila de eventos para indicar os eventos que foram observados;
- gera novos desejos possíveis a partir dos planos encontrados com eventos compatíveis na lista de eventos;
- seleciona deste conjunto de planos combatíveis um evento para execução;
- coloca o evento escolhido em uma pilha nova ou já existente de intenção, dependendo se o evento é ou não um subobjetivo;
- seleciona uma pilha de intenção, pega o plano do topo e executa o próximo passo do plano corrente: se o passo for uma ação, executa a ação, senão, se for um subobjetivo o envia para a fila de eventos.

C.JACK

JACK Intelligent Agents ™ é um *framework* em Java para o desenvolvimento de sistemas multi-agentes, construído pela *Agent Oriented Software Pty. Ltd*.. Possui uma linguagem de desenvolvimento, denominada *JACK Agent Language*, que estende as funcionalidades de Java para a orientação de agentes. Suporta a criação de classes, definições e comandos orientados a agentes. A linguagem JACK permite o desenvolvimento de componentes necessários aos agentes BDI e ao seus comportamento, representados pelas unidades funcionais:

- *Agent*: responsável por definir o comportamento de um agente de software inteligente;
- *Capabibility*: responsável por representar os aspectos funcionais de um agente;
- *BeliefSet*: representa as crencas do agente;
- *View*: permite que consultas de propósito geral possam ser feitas sobre o modelo de dados;
- *Event*: responsável por descrever uma ocorrência que o agetne deve tomar uma ação como resposta;
- *Plan*: são os planos do agente, semelhantes às funções. O que o agente segue para atingir seus objetivos e tratar os eventos.

JACK possui uma plataforma de desenolvimento denominada *JACK Development Environment* (JDE) que permite a criação das unidades funcinais, gerando o código automaticamente. Possui uma ferramenta de design (*Design Tool*) e um Editor de Planos Gráficos.

D.Outras Linguagens

Outras linguagens linguagens foram desenvolvidas para implementar agentes com raciocínio, entre elas podemos destacar:

Jadex é um mecanismo de raciocínio orientado a agentes, desenvolvido por Pokahr, Braubach e Lamersdorf, em que os agentes são desenvolvidos em XML e Java [Pokahr 2003]. A principal característica do Jadex é que ele não é uma nova linguagem, os agentes podem ser escritos nos ambientes de desenvolvimento integrado orientado a objetos.

JAM é uma arquitetura de agentes inteligentes implementada em Java, desenvolvida por Hubber [Hubber 1999]. Foi construída com base em uma série de teorias sobre agentes inteligentes, com enfoque para o modelo BDI. Cada agente é construído por cinco componentes primários: um modelo do mundo (*world model*), uma biblioteca de planos (*plan library*), um interpretador (*interpreter*), uma estrutura de intenções (*intention structure*) e um observador (*observer*).

JASON é uma plataforma de desenvolvimento de sistemas multi-agentes desenvolvida por Bordini, Wooldridge e Hübner [Bordini 2007]. É baseada em um interpretador para uma versão estendida da linguagem AgentSpeak(L). Possui um ambiente de desenvolvimento integrado permitindo a execução e depuração dos agentes.

E.Comparações

Nunes [Nunes 2007] faz uma comparação das principais implementações da arquitetura BDI. JACK não é uma plataforma livre, mais voltada para as aplicações da indústria. Jadex diferencia-se por não requerer o uso de uma nova linguagem. A linguagem Jason possui uma semântica formal, permitindo a verificação formal dos programas. Essa comparação é apresentada nas tabelas a seguir:

	Linguagem
JACK	JACK (extensão de Java)
Jadex	Java e XML
JAM	JAM (extensão do Java)
Jason	Agent Speak (L)

Tabela 1: Linguagens

	Ferramentas de Desenvolvimento
JACK	IDE e Debug
Jadex	Ferramentas para execução, debug e documentação
JAM	-
Jason	IDE e Mind Inspector

Tabela 2: Ferramentas de Desenvolvimento

	Aplicações Comerciais
JACK	UVAs, Tráfego Aéreo e Real-time Scheduling
Jadex	MedPage, Dynatech e Bookstore
JAM	-
Jason	-

Tabela 3: Aplicações Comerciais

III. Conclusão

O estudo de arquiteturas de agentes deliberativos baseados em estados mentais surgiu da necessidade do uso desses agentes no desenvolvimento de aplicações de tempo real em ambientes dinâmicos complexos. A arquitetura BDI permite modelar agentes com estados mentais: crenças, desejos e intenções, possibilitando o desenvolvimento de agentes com raciocínio. Foram desenvolvidos vários *frameworks* e linguagens que de alguma forma capturam a essência dos conceitos da arquitetura BDI, permitindo a modelagem e implementação de sistemas de agentes com raciocínio, como JACK, Jadex, JAM e Janson. Nesse trabalho foram apresentados os princípios do Modelo BDI e da arquitetura BDI e uma comparação de algumas implementações.

Referências

[Bordini 2007] Bordini, R.H., Wooldridge, M. & Hübner, J.F. - Programming Multi-Agent Systems in AgentSpeak using Jason (Wiley Series in Agent Technology).

[Bratman 1987] Bratman, M.E. - Intention, Plans and Practical Reason. Havard University Press, Cambridge.

[D´Inverno 1997] D`Inverno, M.; Kinny, D., Luck, M. & Wooldridge, M. J. - A formal specification of dMARS. In Agent Theories, Architectures, and Languages, pp. 155-176.

[Huber 1999] Huber, M.J. - JAM: A bdi-theoretic mobile agent architecture. In Agents´99: Proceedings of the third annual conference on Autonomous Agents.

[Nunes 2007] Nunes, I.O. - Implementação do Modelo e da Arquitetura BDI. Monografia em Ciências da Computação, PUCRJ.

[Pokahr 2003] Pokahr, A., Braubach, L. Lamersdorf, W. - Jadex: Implementing a BDI – Infrastructure for JADE Agents – EXP in Search of Innovation.

[PRS] Procedural Reasoning System – Wikipedia. Disponível em <http://en.wikipedia.org/wiki/Procedural_reasoning_system> acesso em 22 de dezembro de 2014.

[Rao 1995] Rao, A., Georgeff, M. - BDI-agents: from theory to practice. In Proceedings of the First Intl. Conference on Multiagent Systems, San Francisco.

[Weiss 1999] Weiss, G. - Multiagent system – a modern approach to distributed artificial intelligent. MIT Press, Cambridge.

[Wooldridge 1995] Wooldridge, M. J.; Jennings, N. R. - Intelligent Agents: Theory and Practice - Knowledge Engineering Review.

[Wooldridge 2000] Wooldridge, M. J.; Jennings, N. R. - Reasoning about Rational Agents. London: The MIT Press.