Abordagem de integração de sistema controle multiagente

Elmer Rolando Llanos Villarreal, [1]
Leonardo Múrcio
Marcelino Pereira dos Santos Silva

Universidade do Estado do Rio Grande do Norte
Departamento de Informática, Telefone/FAX: (084) 3315-2235
BR 110, Km 48, Bairro Costa e Silva , Mossoró RN.
e-mail: evillarrea@hotmail.com; leonardo.murcio@hotmail.com; marcelinopereira@uern.br;

Abstract

Resumo. No presente artigo apresenta-se uma abordagem de integração de controladores com uma plataforma multiagentes, de tal forma de integrar o SMA em linguagem $KQML$ com o sistema de simulação de uma planta. Assim elabora-se uma modelagem para construção de uma plataforma, para finalmente avaliar a resposta do sistema integrado. Assim apresenta-se proposta de um sistema multiagente , destinado ao monitoramento e controle de um determinado processo industrial, assim como comunicação entre o modelo dinâmico e o SMA em linguagem $KQML$.
Palavras-chave: Sistemas de Controle, Plataforma multiagente, Linguagem $KQML$.

1 Introdução

Os sistemas modernos de computação frequentemente envolvem múltiplos computadores interagindo de forma distribuída em [5]. Neste sentido, a habilidade de comunicação é um importante atributo que os agentes inteligentes devem possuir. Ambientes que possuem mais de um agente, praticamente exigem o intercâmbio de informações. Neste sentido, torna-se claro a necessidade de uma linguagem de comunicação comum. Apesar desta afirmação ser clara e bem aceita pela comunidade de pesquisadores, o problema é definir qual a linguagem ideal. Atualmente, não existe uma linguagem padronizada e aceita mundialmente para a representação de informações trocadas por agentes. A presença de uma linguagem comum para troca de mensagens é uma das características que diferencia um agente de um objeto na programação orientada a objetos. Uma mensagem no contexto dos agentes carrega uma semântica independente do agente, enquanto que uma mensagem no contexto dos objetos pode variar de um objeto para outro.

Em [4] estabelecem que existem diversos níveis nos quais sistemas baseados em agentes devem interagir:

- Transporte: como agentes enviam e recebem mensagens.
- Linguagem: qual o sentido de mensagens individuais.
- Política: como os agentes estruturam conversações.
- Arquitetura: como conectar sistemas em concordância com protocolos existentes.

O projeto de uma linguagem de comunicação entre agentes utiliza normalmente uma das seguintes abordagens: procedural ou declarativa. Na abordagem procedural, a comunicação acontece através de diretivas. Tanto comandos individuais quanto programas completos podem ser transmitidos e executados no lado receptor. Linguagens baseadas em scripts, tais como TCL e Telescript, são exemplos da abordagem procedural.

Na abordagem declarativa, a comunicação ocorre através da troca de estruturas declarativas, tais como definições, asserções e outras. A abordagem declarativa pode ser encontrada em [8], que escolhem a Linguagem de Comunicação entre Agentes (ACL - Agent Communication Language) como a base de comunicação para sua plataforma de Engenharia de Software baseada em agentes. A ACL foi o resultado do grupo $ARPA\ KSE$ (Knowledge Sharing Effort) [13]. Pode-se dividir a ACL em três partes:

- Vocabulário: funciona como um grande dicionário de palavras apropriadas às áreas de aplicação comum. Este vocabulário compartilhado é chamado de ontologia.
- Linguagem interna: KIF (Knowledge Interchange Format) é uma versão do cálculo de predicados de primeira ordem. Possui a capacidade de codificar dados simples, restrições, regras, expressões e outras. Uma completa descrição da linguagem KIF e de como utilizá-la é encontrado [7].
- Linguagem externa: $KQML$ (Knowledge Query and Manipulation Language) é uma camada lingística que pode encapsular estruturas KIF. Assim ela fornece informação contextual para uma comunicação mais eficiente. Uma das especificações iniciais de $KQML$ em [4].

Uma mensagem ACL é uma expressão $KQML$ na qual os argumentos são termos ou sentenças KIF formadas por palavras no vocabulário ACL. Entretanto, é importante ressaltar que a linguagem $KQML$ não foi idealizada para transmitir apenas um conteúdo baseado em KIF. Na verdade, $KQML$ pode ser bem mais abrangente, permitindo até a transmissão de um conteúdo definido pelos próprios projetistas do sistema multi-agente em questão. Obviamente, a não utilização de uma linguagem padronizada e com habilidade de representação de conhecimento, como KIF, reduz a possibilidade de interação entre agentes desenvolvidos por projetistas diferentes.

O artigo é organizado como segue. Na seção (2) são apresentados os conceitos básicos de Linguagem $KQML$. Na seção (3) são apresentados as arquiteturas de agentes. Na seção (4) são apresentados as especificações do controle. Na seção (5) são apresentados a arquitetura de sistema de controle multiagente. Na seção (6) são apresentados a comunicação entre um modelo dinâmico e o SMA em linguagem $KQML$.Finalmente apresentado uma integração de controladores e as conclusões finais.

2 Linguagem KQML

2.1 Conceitos básicos

A linguagem $KQML$ fornece uma plataforma para programas e agentes trocarem informações e conhecimento. $KQML$ está focada nos formatos de mensagem e em protocolos de manipulação destas mensagens entre agentes em execução. Entretanto, $KQML$ não se preocupa com o formato da informação propriamente dita. Suas expressões usualmente encapsulam estruturas de outras linguagens denominadas "linguagens de conteúdo". $KQML$ é uma linguagem que permite programas realizarem operações sobre as bases de conhecimento de cada agente envolvido.

É importante notar que, por se tratar de uma linguagem, alguns termos apresentados neste artigo não serão traduzidos. Uma mensagem $KQML$ é chamada performative. Cada mensagem tem o objetivo implícito de realizar alguma ação específica. Como pode ser observado na especificação da linguagem [4], existe um grande número de performatives definidas e a maioria dos sistemas baseados em agente suportam somente um pequeno subconjunto delas. As performatives, ou tipos de mensagem, são palavras reservadas em $KQML$. Usando performatives, agentes podem perguntar a outros agentes por informações, dizer a outros agentes fatos, divulgar seus serviços a outros agentes e solicitar serviços de outros agentes.

$KQML$ adota o uso de ontologias. Ontologias são um conjunto de especificações explícitas de significado, conceitos e relacionamentos aplicáveis a algum domínio específico. Desta forma, pode-se assegurar que dois agentes estejam utilizando a mesma linguagem durante o processo de comunicação. Em outras palavras, a utilização de uma ontologia permite a definição de um contexto único, eliminando-se a ambigidade.

Mensagens $KQML$ codificam informação em três diferentes níveis arquiteturais: conteúdo, mensagem e comunicação. Sua sintaxe está baseada na linguagem *Lisp* e é composta de uma ação (performative) e parâmetros. A ordem de posicionamento dos parâmetros não é importante. Os parâmetros são codificados como pares ¡palavra-chave valor¿, onde a palavra-chave é precedida pelo símbolo ":".

2.2 Características da linguagem KQML

Uma característica interessante da linguagem $KQML$ é que ela pode oferecer uma forma de acesso à informação, mesmo para programas que não sejam agentes. Em [4], a implementação de um agente não é necessariamente estruturada como sendo uma base de conhecimento. Sua implementação pode ser um simples sistema de banco de dados ou um programa utilizando um estrutura de dados especial encapsulada por uma interface de acesso semelhante a uma base de conhecimento. Isto permitiria que outros agentes tivessem acesso às informações disponíveis. Desta forma, cada agente poderia gerenciar uma base de conhecimento virtual (VKB Virtual Knowledge Base).

3 Arquiteturas de Agentes

Uma arquitetura de software pode ser descrita como sendo a configuração dos componentes que constituem um sistema e das conexões que coordenam as atividades entre estes componentes [1]. Entretanto, uma arquitetura para agentes refere-se ao modo de organização dos agentes dentro de um sistema e como estão estruturados seus relacionamentos e interações.

Assim como existem diversas arquiteturas de software, o mesmo ocorre com relação as arquiteturas de agentes, as quais possuem certas características que permitem a avaliação de sua qualidade e eficácia [6].

Em [15], uma arquitetura de um agente pode ser estruturada através de uma metodologia específica para definir agentes. Sendo assim, a arquitetura abrangeria técnicas e algoritmos para suportar esta metodologia.

Segundo [10], a discussão sobre a qualidade de uma arquitetura de agentes torna-se subjetiva, uma vez que os detalhes desta discussão dependem de aspectos específicos da aplicação agente que se pretende desenvolver. No entanto, em [12] procura-se estabelecer alguns conceitos que podem ser úteis para o desenvolvimento de uma arquitetura promissora:

- Simplicidade: idealizar a arquitetura e seus componentes de forma que sejam fáceis de entender, implementar e manter.
- Funcionalidade: selecionar uma arquitetura e ferramentas de desenvolvimento que focalizem aspectos específicos do problema a ser abordado.
- Expansividade: a arquitetura deve poder ser ampliada, uma vez que nem todas as necessidades futuras podem ser previstas em um primeiro momento.
- Isolamento ou Portabilidade: uma arquitetura, para poder ser expansiva, deve possuir uma implementação portável, evitando-se soluções não padronizadas.

3.1 Classificação de Arquiteturas

O termo arquitetura pode compreender uma faixa razoável de possibilidades, principalmente no aspecto complexidade. Arquiteturas de agentes não são uma exceção e podem ser classificadas de acordo com as necessidades da aplicação, dos usuários, e o grau de sofisticação ou nível de inteligência dos agentes. De acordo com [10], a complexidade de uma arquitetura pode ser classificada em três grupos:

- Arquitetura simples: quando é composta por um único e simples agente.
- Arquitetura moderada: quando é composta por alguns agentes que realizam as mesmas tarefas, mas possuem diferentes usuários e podem residir em máquinas diferentes.
- Arquitetura complexa: quando é composta por diferentes tipos de agentes, cada um com certa autonômia, podendo cooperar e estar em diferentes plataformas.

Em [15] se baseiam na forma de construção dos agentes envolvidos para dividir as arquiteturas em três áreas:

- Arquitetura deliberativa: segue a abordagem clássica da Inteligência Artificial, onde os agentes atuam com pouca autonômia e possuem modelos simbólicos de seus ambientes, é dizer, esta arquitetura interpreta os agentes como parte de um sistema baseado em conhecimento.
- Arquitetura reativa: esta abordagem procura não utilizar nenhum tipo de modelo ou raciocínio simbólico. Este tipo de arquitetura baseia-se na proposta que um agente pode desenvolver inteligência a partir de interações com seu ambiente, não necessitando de um modelo pré-estabelecido.
- Arquitetura híbrida: como o próprio nome sugere, este tipo de arquitetura combina características das duas abordagens anteriores.

3.2 Arquitetura Genesereth

O artigo em [8] está focado na interoperabilidade de software. A proposta foi facilitar a criação de sistemas com habilidade de interação através de uma engenharia de software baseada em agentes. Nesta abordagem, os programas de aplicação são escritos como sendo agentes de software. Estes agentes podem ser interpretados como componentes que comunicam-se com seus pares por meio da troca de mensagens através de uma linguagem de comunicação de agentes. Apesar da semelhança com objetos, os agentes utilizam uma linguagem comum com uma semântica independente de agente enquanto que mensagens entre objetos podem variar de objeto para objeto. Esta arquitetura adota a abordagem ACL (Agents Communication Language), para a comunicação entre os agentes.

Outra questão também tratada em [8] foi a preocupação com os programas já existentes. Sendo assim, eles proporam três abordagens para o que denominaram agentificação:

- Implementar um programa tradutor que atua como um mediador entre o programa e outros agentes.
- Implementar uma camada adicional para o programa existente, provendo este com a capacidade de se comunicar via ACL.
- Reescrever o programa original. Esta proposta deve ser utilizada apenas em último recurso.

Uma vez que a arquitetura tem definida sua plataforma de comunicação, torna-se importante entender como os agentes estão organizados. Existem duas abordagens diferentes: comunicação direta e coordenação assistida. Na comunicação direta, a coordenação é gerenciada pelo próprio agente. Na coordenação assistida, o agente delega a atividade de coordenação à programas especiais.

A principal vantagem da comunicação direta está na não dependência de outros programas. Duas arquiteturas conhecidas que implementam a comunicação direta são as redes de contrato (contract net) e compartilhamento de especificação (specification sharing). Nas redes de contrato, um agente solicita que outros agentes submetam suas propostas, os quais declaram-se para a realização de certa atividade. O agente recebe e analisa as declarações recebidas e estabelece um contrato com agente mais promissor. O compartilhamento de especificação usa a abordagem inversa, onde os agentes divulgam seus interesses e capacidades para os demais agentes. Neste caso, o número de mensagens trocadas é menor que nas redes de contrato, pois não há necessidade de haver sempre uma negociação.

O maior problema destas abordagens é a quantidade de tráfego gerado, uma vez que são soluções baseadas na difusão de mensagens. A complexidade do processo de negociação também é repassada aos agentes. Sendo assim, é proposta uma solução baseada em comunicação indireta denominada sistema federado.

No sistema federado, a interação dos agentes é assistida por um programa especial denominado facilitador, o qual oferece um conjunto de serviços de coordenação [9].

Os agentes utilizam uma estrutura semelhante ao compartilhamento de especificação, mas apenas com seu facilitador. O facilitador atua então como um mediador, roteando mensagens (solicitações e respostas) de acordo com seu conhecimento interno.

A principal característica dos facilitadores que o diferenciam de simples roteadores de mensagens (brokers) está associada à sofisticação de processamento. Um facilitador pode utilizar a documentação das necessidades e capacidades de um agente expressada através da ACL para melhorar o processo do roteamento de mensagens.

4 Especificação de controle

As especificações de controladores de malha são maiormente apresentado por equações e representações gráficas. A especificação provee informação de fluxo de dados e transição de estado do controlador [14]. Para o framework do controlador multiagente em [2] . Assim baseado no diagrama organizacional que é proposto para a estrutura hierarquica ilustrada dos agentes nosso estudo esta sendo direcionado.

Assim por exemplo no artigo em [2] elaborou-se um Framework para projetar e implementar sistemas de controladores com estrutura hierarquica usando um Framework baseado em agente. Assim baseado neste framework, e no conceito de agente-controlador foi realizado a pesquisa, No framework introduzido contém todas as informações de um subproblema de controle particular e uma agênte-controlador, o qual consiste de um pool de agentes-controladores e um objeto de coordinação. Assim o objeto de coordinação é responsável por resolver conflitos, ordem execução sequêncial e ações que misturam um pool de agentes-controladores dentro de uma agência-controlador. Diferentes tipos de mecanismos de coordinação tal como o paralelo, prioridades fixada, sequencial, adição tem sido descrito.

O instrumento, "Controlador Multiagente", para o projeto integrado e Instrumento de Implementação, desenvolvido neste projeto, é baseado em um Framework de Implementação de Controlador Multiagente". Sistemas de controle envolve sensores, o qual mede dados da planta e atuadores, que controla a planta de acordo com as referncias dadas. Do ponto de vista do controlador, o controlador tem dados e referncias como entrada e sinais de atuação como saídas . O controlador multiagente consiste de um agente singular denominado "agente principal" o qual é constituido de diferentes agentes para a leitura de entrada de dados (sensores e referências), processando o dado de entrada (algoritmo de controle) e saindo o sinal de atuação (atuador). Os agentes junto com o agente principal são responsavéis pelas tarefas locais e controle da planta via comunicação entre cada outro. Seis componentes primarias diferentes constitui a construção de blocos do MACIF. Esses agentes são brevemente descritos como segue.

4.1 Componentes

- AgenteSensor: Adquire dados do ambiente e expediciona os dados para outros agentes. O ambiente envolve a planta a ser controlado e outros sistemas externos tais como Interfaces Humanas (da referencias) ou disturbios. Na implementação em tempo discreto, os agentes sensores adquire essos dados de AD-conversores (o qual adquire dado do ambiente).

- Agentecomposto: consiste de um grupo de agentes controladores (i.e. Agentes elementares e/ou outros agentes compostos) e a coordenação objeto.

- Agente principal (Controlador multiagente) é um agente total responsável pelo sistema de controle inteiro. O agente principal é um bom ou melhor do agente composto assim consiste de um grupo de agentes controladores e uma coordenação objeto. Em adição o agente principal

também consiste de um grupo de agentes sensores e agentes atuadores que relaciona com o ambiente. O agente principal é literalmente um agente principal consegue informação da planta para ser controlada por seus sensores agentes. O sistema de controle multiagente tem unicamente um agente principal, que é responsável por relacionar com o sistema físico (planta controlada).

- Coordenação objeto:esta contida em todos os agentes compostos (e principalmente agentes). A tarefa de uma coordinação de objeto é coordenar o comportamento do agente controlador que existam com um agente composto(ou um agente principal).
- Agente elemental: é um agente fundamental do $MACIF$ e implementa soluções de controle local do problema de controle global.

5 Arquitetura de sistema de controle multiagente

5.1 Sistema controlador multiagente

A arquitetura da implementação de agentes varia dependendo da descrição do problema e sua abordagem de solução. Além disso o príncipio básico de agentes autônomos com capacidade de comunicação é elementar em quase todas as implementações. Assim para implementar o conceito de agente baseado em engenharia de controle nosso estudo foi baseado na proposta de [2]. O framework para implementar o controlador multiagente (MACIF) apresenta backbone genérico para implementar controladores como agentes. O framework introduz agentes controladores, os quais podem ser integrados e eficientemente coordinados para resolver problemas de controle complexo. O agente controlador pode ser costurado para implementar controladores requeridos. Assim nesse estudo o framework também apresenta um guideline para implementar esses agente-controlador para resolver problemas de controle geral e estamos aprofundado nosso estudo nesta parte. O instrumento, Controlador Multiagente, para o projeto integrado e Instrumento de Implementação, desenvolvido neste projeto, é baseado em um Framework de Implementação de Controlador Multiagente. Sistemas de controle envolve sensores, o qual mede dados da planta e atuadores, que controla a planta de acordo com as referências dadas. Do ponto de vista do controlador, o controlador tem dados e referências como entrada e sinais de atuação como saídas . O controlador multiagente consiste de um agente singular denominado agente principal o qual é constituido de diferentes agentes para a leitura de entrada de dados (senores e referências), processando o dado de entrada (algoritmo de controle) e saindo o sinal de atuação (atuador). Os agentes junto com o agente principal são responsavéis pelas tarefas locais e controle da planta via comunicação entre cada outro.

Assim por exemplo se o objetivo é a proposta de um sistema multiagente, destino ao monitoramento da produção e controle de um determinado processo industrial, permitindo que cada um dos agentes que compõem esta sociedade atue de uma forma eficiente obtendo melhores resultados na produção e controle da planta insdustrial. Assim neste caso específico, deve-se ter os seguintes objetivos:

1. Desenvolver o modelo dinâmico onde será testado o Sistema Multiagente (SMA).
2. Desenhar o sistema multiagente definindo o tipo de mensagem, protocolos de interação e comportamento de cada um dos agentes.
3. Desenvolver um protótipo funcional do Sistema multiagente baseado nas especificações obtidas na fase do desenho [3].
4. Integrar o SMA com o sistema de simulação.
5. Avaliar a resposta do sistema integrado.

6 Comunicação entre o modelo dinâmico e SMA em KQML

6.1 Comunicação entre plataformas

A comunicação entre a plataforma de Agentes $KQML$ e o software de simulação SIMULINK deve ser através do uso do protocolo TCP/IP. O TCP/IP é o protocolo de rede mais usado atualmente. O TCP especifica o formato dos pacotes de dados e reconhecimentos que dois computadores trocam para realizar uma transferência confiável, assim como os procedimentos que os computadores usam para assegurar que os dados cheguem corretamente. Assim pode-se ver o TCP como um "duto" entre dois computadores ligados a duas portas. Para ter uma comunicação via TCP/IP necesita-se de um Cliente e de um Servidor. O servidor é o extremo da comunicação que inicia a comunicação utilizando uma porta da rede como entrada. O Cliente procura esta porta e se conecta ao "Servidor". Assim são analizados diferentes tipos de protocolos $FIPA$ e os protocolos de comunicação [11].

7 Abordagem integrada para controladores

Assim neste artigo é baseado no desenvolvimentos significantes no campo de arquitetura de controladores que providência um suporte suficiente para simplificar o projeto de sistemas de controle complexos. Pois o instrumento projeto integrador e instrumento de implementação para controladores multiagentes (IDITMAC) é baseado no framework de controlador multiagente em [2]. Assim o instrumento é pode ser primeiramente projetado para combinar uso com SIMULINK(projeto de controle) e $KQML$ (implementação na comunicação de agentes) O projeto controlador pode ser simulado com $SIMULINK$ via uma libraria (toolbox) e implementado na plataforma para comunicação de agentes na Linguagem $KQML$.

8 Conclusões

No presente artigo apresentou-se na primeira parte um estudo da linguagem $KQML$, assim como as arquiteturas de agentes, com o objetivo de elaborar uma arquitetura de sistema de controle multiagente. Na segunda parte apresentou-se uma abordagem de integração de controladores multiagentes com uma plataforma, de tal forma de integrar o SMA em linguagem $KQML$ com o sistema de simulação. Assim elaborou-se uma modelagem para construção de uma plataforma, para finalmente avaliar a resposta do sistema integrado. Assim apresenta-se proposta de um sistema multiagente, destinado ao monitoramento e controle de um determinado processo industrial. Assim também apresentou-se como deve ser a comunicação entre o modelo dinâmico e o SMA em linguagem $KQML$.

References

[1] G.Abowd,, J. Engeslma, L. Guadagno e O. Okon, *Architectural Analysis of Object Request Brokers*, Object Magazine, March, pp. 44-51, 98, 1996.

[2] A. J. N. Van Breemen, "Agent-Based Multi-Controller System", University of Twente Press, Enschede, The Netherlands, NL. 2001.

[3] E. R. LL. Villarreal, M. P. S. Silva, D. P. F. Pedrosa and J. D. Lima *Abordagem de integração do sistema computacional para redução de perdas em redes de distribuição de energia com uma plataforma de controle multiagente*, Aceito Revista de Ciência Sempre, FAPERN, Natal, RN, 2008.

[4] T. Finin, J. Weber, J. Draft: Specification of KQML Agent Communication Language. The DARPA Knowledge Sharing Initiative External Interfaces Working Group, *http : //www.cs.umbc.edu/kqml/kqml.ps*, June 1993.

[5] T. Finin, R. Fritzson, D. McKay, and R. McEntire, *KQML as an Agent Communication Language*, Proceedings of the 3rd International Conference on Information and Knowledge Management, ACM Press, November, 1994.

[6] Guillerme Bittencourt, *Inteligência Artificial Distruída*, I Workshop de Computação, Instituto Tecnológico de Aeronáutica, São José dos Campos, SP, outubro, 1998.

[7] Genesereth, M. e Fikes, R. Knowledge Interchange Format, Version 3.0 Reference Manual. Logic Group, Computer Science Department, Stanford University. June 1992.

[8] Genesereth M. R. e Ketchpel, S. P. Software Agents. Communications of the ACM, July 1994, pp. 48-53, 147.

[9] Jennings, N. R., *Cooperation in Industrial Multiagent Systems*, World Scientific, 1994.

[10] M. Knapik, J. Johnson, *Developing Inteligent Agents for Distributed Systems*, Computing McGraw-Hill, NY:McGraw-Hill, 1998.

[11] M. Marietto. , G. Bruno, D. Nuno, S. J. Simão and C. Helder,*Requeriments Analysis of Multi-Agent-Based Simulation Platforms*, Relatório Técnico, Universidade de São Paulo, 2002.

[12] T. Mowbray, *Essentials of Object-Oriented Architecture*, Object Magazine, September 1995, pp. 28-32.

[13] R. Neches, R. Fikes, T. Finin, T. Gruber, R. Patil, T. senator, and W. swartout, *Enabling Technology for Knowledge Sharing*, AI Magazine, Vol. 3, No. 12, pp.36-56, 1991.

[14] Ogata, *Engenharia de Controle Moderno*,Prentice Hall 2004.

[15] M. Wooldridge, e N. R. Jennings, *Intelligent Agents: Theory and Practice*, Submitted to the Knowledge Engineering Review, 1994.