Controle de Conflitos em Documentos Ativos Multiagentes

Flávio Miguel Varejão, Rodrigo Laiola Guimarães

Departamento de Informática – Universidade Federal do Espírito Santo (UFES)
Av. Fernando Ferrari, S/N - Vitória - ES - Brazil

fvarejao@inf.ufes.br, rodrigolaiola@bol.com.br

Abstract. *This paper describes how MultiJADE, a domain independent multiagent active documents environment, identifies and supports conflict mitigation in a concurrent and distributed problem solving process.*

Resumo. *Este artigo descreve como MultiJADE, um ambiente multiagente baseado em documentos ativos independente de domínio, identifica e dá suporte a mitigação de conflitos em um processo de resolução de problemas concorrentes e distribuídos.*

1. Introdução

Uma abordagem que tem sido cada vez mais investigada procura realizar uma parceria entre agentes computacionais, ditos inteligentes, e agentes humanos para a realização da tomada de decisão. Nesta linha surgiram os Documentos Ativos de Design (Active Design Documents ou ADD) [Garcia e Howard 1993]. Sistemas ADD capturam a informação sobre as decisões durante o processo de design em um domínio específico e não lidam com problemas de design concorrente e distribuído. Através da integração da abordagem ADD com o conceito de sistemas multiagentes surgiu o MultiADD (Multiagent Active Design Documents) [Garcia e Vivacqua 1997], um sistema capaz de suportar trabalho colaborativo em problemas de design concorrente e distribuído. MultiJADE (Multiagent Java-based Active Documents Environment) [Gama *et al.* 2004] é um ambiente multiagente que generaliza a abordagem MultiADD para outras tarefas além de design e que torna a construção de sistemas MultiADD em diferentes domínios mais simples.

Este artigo apresenta a arquitetura do dispositivo responsável pela identificação e notificação de conflitos do ambiente MultiJADE, e encontra-se organizado como a seguir. Na seção 2 são apresentados os conceitos de Documentos Ativos de Design e da abordagem MultiADD. Por sua vez, a seção 3 introduz o ambiente MultiJADE. A seção 4 destaca a arquitetura do servidor controlador, dispositivo que identifica e controla os conflitos no MultiJADE. Por fim, a seção 5 se dedica às conclusões e considerações finais.

2. Documentos Ativos de Design

Sistemas ADD são Sistemas Baseados em Conhecimento para um domínio particular que utilizam a metáfora da existência de um aprendiz computacional que acompanha o responsável humano na resolução do problema, apoiando-o e registrando automaticamente suas decisões. A interação entre o ser humano e o aprendiz ocorre de duas maneiras: o ser humano pode pedir ao aprendiz que realize cálculos e tome decisões, ou pode ele mesmo realizar essas tarefas e requerer que o aprendiz as acompanhe. Neste último caso, para cada decisão tomada, o sistema compara a decisão do projetista com a decisão que seria tomada segundo o conhecimento armazenado. Se houver diferenças, o sistema interroga o usuário para que este possa alterar sua decisão ou ajustar a base de conhecimento do sistema.

2.1. JADE

Sistemas ADD são dependentes de domínio e sempre que um Documento Ativo de Design é necessário em um novo domínio, um sistema precisa ser construído do zero. Varejão e Pessoa (2002) desenvolveram o JADE (Java-based Active Documents Environment), um ambiente computacional para construção de sistemas ADD independente de domínio e de tarefa. Além disso, JADE aumenta os tipos de interação entre os seres humanos e o sistema e torna o ambiente computacional ADD mais fácil de ser customizado e estendido. JADE amplia a capacidade de interação dos sistemas ADD, permitindo o usuário final modificar mais facilmente os parâmetros e a topologia da rede de parâmetros. Além disso, JADE ameniza o esforço e o custo requerido para construir e mudar sistemas ADD, uma vez que o usuário utiliza a mesma interface de aquisição para criar e modificar a rede de parâmetros.

2.2. MultiADD

Na arquitetura MultiADD um agente é constituído do sistema computacional ADD e do projetista. Cada sistema ADD possui sua rede paramétrica específica para representar sua parte do conhecimento dentro do contexto global. Mesmo que na maior parte do processo de design um agente trabalhe independentemente dos demais, podem existir dependências entre suas decisões e as decisões de outros agentes. Se surgirem conflitos, os agentes devem interagir para que uma boa solução global seja alcançada. Além dos agentes especialistas, MultiADD possui um agente coordenador responsável por supervisionar o processo de mitigação de conflitos e resolver eventuais impasses. A figura deste agente garante que um conflito não vai persistir por um período que possa comprometer o andamento do processo de resolução do problema. Existe ainda um sistema de controle responsável por identificar e comunicar a ocorrência de conflitos aos agentes envolvidos e ao coordenador.

3. MultiJADE

O ambiente computacional JADE não trata problemas concorrentes e distribuídos. Gama et al. (2004) desenvolveram o MultiJADE, uma extensão do JADE com características da abordagem MultiADD. Para modelar o problema concorrente, além dos parâmetros da modelagem ADD surge no MultiJADE o conceito de “Parâmetro de Comunicação”. A dependência entre estes parâmetros nos agentes é definida através de regras agregadas pelo engenheiro de conhecimento durante a construção da base de conhecimento compartilhada pelos agentes. Os parâmetros de comunicação são sempre utilizados por dois ou mais agentes.

4. Identificação de conflitos no MultiJADE

O controlador é responsável pela comunicação entre agentes, identificação de conflitos e suporte a sua resolução. No MultiJADE, os agentes não se comunicam diretamente entre si, eles enviam mensagens para o controlador que repassa as informações convenientes para os demais agentes na ordem e momento apropriado.

Na Figura 1 é apresentada a arquitetura do servidor controlador. O módulo de acesso é responsável pela comunicação entre agentes e servidores. Por este módulo, os agentes podem tanto se comunicar com os servidores quanto receber mensagens de notificação relativas a conflitos.

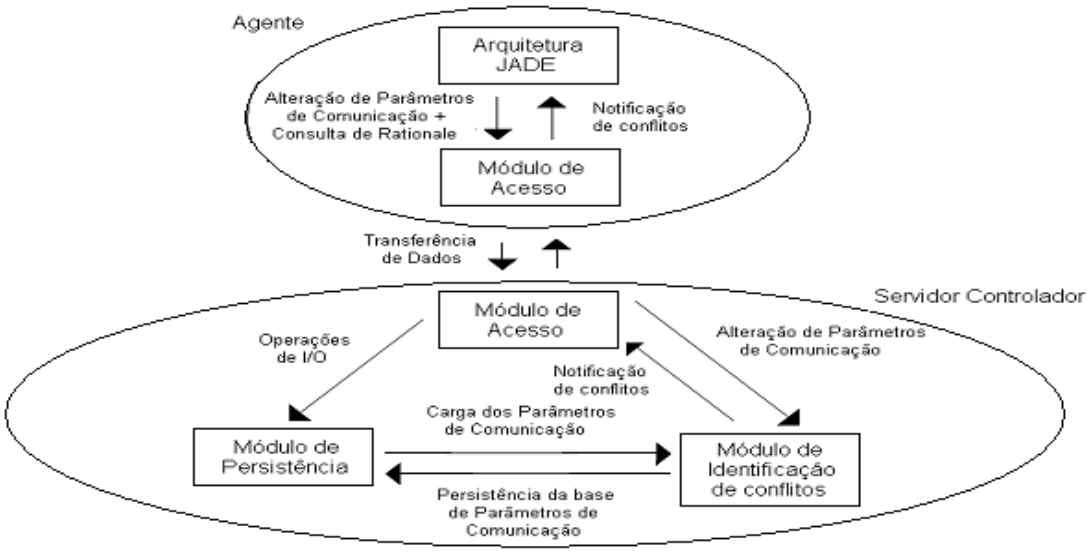

Figura 1. Arquitetura do servidor controlador

O módulo de persistência é responsável pelo armazenamento e gerência de todos os dados de um projeto, dentre eles, os documentos ativos dos agentes. O módulo de identificação de conflitos é responsável pela identificação e suporte a todo processo desde o surgimento até o término de um conflito.

Na criação de um projeto, uma instância da base de conhecimento é armazenada no módulo de persistência. Neste momento, também são registrados no módulo de identificação de conflitos todos os parâmetros de comunicação bem como suas relações. Este registro em uma outra base ocorre por dois motivos. O primeiro se deve ao fato de que se um usuário está trabalhando em seu documento ativo, a versão mais atual ainda não está no servidor controlador. Desta forma, o servidor controlador não seria capaz de identificar um conflito imediatamente após a modificação de um parâmetro de comunicação visto que poderia não possuir a versão mais atualizada de todos os documentos ativos. Em segundo lugar, seria inviável atualizar um documento ativo no servidor controlador a cada modificação de um parâmetro de comunicação para depois abrí-lo em busca de conflitos.

Quando um parâmetro de comunicação é modificado em um agente, uma mensagem é enviada ao servidor controlador através do módulo de acesso. Recebendo esta mensagem, o módulo de acesso do servidor controlador a encaminha ao módulo de identificação de conflitos. É disparado um processo que avalia se as relações em torno do parâmetro de comunicação em questão foram obedecidas. Havendo descumprimento de uma das relações, o módulo de identificação envia uma mensagem de notificação de conflito para cada um dos agentes relacionados àquele parâmetro de comunicação. O envio desta mensagem é realizado através do módulo de acesso do servidor controlador.

Caso um agente queira consultar o *rationale* de outro agente, uma solicitação é feita através do seu módulo de acesso. Ao recebê-la, o módulo de acesso do servidor controlador a encaminha para o módulo de persistência. Como resposta, é enviada uma cópia do documento ativo requerido para que o agente que a solicitou possa consultá-la. Os agentes também podem utilizar ferramentas de correio eletrônico e chat para que o processo de negociação seja facilitado. Todas as mensagens de correio e sessões de chat são automaticamente enviadas ao servidor controlador para uma eventual necessidade

de consulta. O armazenamento desses dados é feito no módulo de persistência e o processo de recuperação acontece da mesma forma que a consulta de *rationale*.

Durante o processo de negociação, os agentes podem modificar o valor dos parâmetros de comunicação. Da mesma forma que foi descrita anteriormente, estas modificações são remetidas ao servidor controlador e em caso de cumprimento das relações daquele parâmetro, o módulo de identificação de conflitos envia uma mensagem de fim de conflito a todos os agentes envolvidos.

O módulo de identificação de conflitos controla o limite de tempo estipulado pelo coordenador para um parâmetro em conflito. Se este tempo expirar é porque o conflito ainda persiste. Então, o módulo de identificação de conflitos encaminha uma mensagem de término do período de negociação a todos os agentes envolvidos e uma mensagem de notificação da necessidade de intervenção ao agente coordenador. Ao intervir, todos os agentes devem acatar a decisão do coordenador e o conflito chega ao fim. O limite de tempo de conflito é especificado pelo agente coordenador e pode ser baseado no impacto que um conflito trará ao desenvolvimento da resolução do problema.

5. Conclusão

MultiJADE utiliza a tecnologia de informação para facilitar a resolução de problemas concorrentes e distribuídos, reduzindo consideravelmente o esforço e o custo requeridos para a construção de sistemas MultiADD de forma a minimizar as barreiras de utilização destes sistemas. No entanto, MultiJADE tem a particularidade de não se propor a ser um solucionador automático de conflitos e sim suportar a resolução dos mesmos. Isso é possível uma vez que MultiJADE ao identificar os conflitos, comunica imediatamente os envolvidos, permitindo que eles tenham maior entendimento sobre as razões das decisões dos demais agentes na hora de negociar um conflito e evitando atrasos decorrentes da necessidade de reuniões presenciais.

6. Referências

Gama, F. A., Varejão, F. M., Guimarães, R. L., Braun, C. S. (2004) *MULTIJADE - a domain independent multiagent active design documents environment,* In: Proceedings of the First International Conference on Design Computing and Cognition - DCC'04, pp. 479-497, Kluwer Academic Publishers, Boston, USA.

Garcia, A. C. B., Howard, H. C. (1993) *Active Design Documents: From Information Archives to Design Model Construction*, In: Proceedings of the Artificial Intelligence in Engineering Conference, pp. 233-244, Toulose, France.

Garcia, A. C. B., Vivacqua, A. (1997) *MultiADD: A Multiagent Active Design Document Model to Support Group Design*, In: Proceedings of the 14th National Conference on Artificial Intelligence, pp. 1066-1071, AAAI Press, USA.

Varejão, F. M., Pessoa, R. M. (2002) *JADE: A Computational Environment For Constructing, Using and Reusing Active Design Documents*, In: Proceedings of the Fifth WG 5.2 Workshop on Knowledge Intensive CAD, pp. 99-114, Malta.