PUC

ISSN 0103-9741

Monografias em Ciência da Computação
n° 11/17

Proveniência de Dados em Sistemas Multiagentes

Tassio Ferenzini Martins Sirqueira
Marx Leles Viana
Carlos José Pereira de Lucena

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO
RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22453-900
RIO DE JANEIRO - BRASIL

Monografias em Ciência da Computação, No. 11/17 ISSN: 0103-9741
Editor: Prof. Carlos José Pereira de Lucena August, 2017

Proveniência de Dados em Sistemas Multiagentes

Tassio Ferenzini Martins Sirqueira[1,2], Marx Leles Viana[1],
Carlos José Pereira de Lucena[1]

[1]Pontificia Universidade Católica do Rio de Janeiro (PUC-Rio)
[2]Instituto Vianna Junior (FIVJ)

tmsirqueira@vianna.edu.br, lucena@inf.puc-rio.br

Abstract. To understand data provenance it is necessary to know the reasoning behind the decisions that lead to this data. Tracking actions on a system is not trivial, especially if it involves intelligent software agents and human beings, where both can make decisions that change the behavior of the system and the results generated by it. This article discusses the data provenance in multiagent systems (MAS), showing a new perspective of the current usage stage of provenance systems, i.e., what benefits the data provenance can bring and the problems involved in the absence of information about the actions of the systems that make use of MAS. In addition, the article mentions some research opportunities that are still untapped in multiagent systems.

Keywords: Data Provenance; Multiagent Systems; Provenance in MAS.

Resumo. Para entender a proveniência dos dados é necessário conhecer o raciocínio por trás das decisões que levam a esses dados. O acompanhamento de ações em um sistema não é trivial, especialmente se envolve agentes de software inteligentes e seres humanos, onde ambos podem tomar decisões que alteram o comportamento do sistema e os resultados gerados por ele. Este artigo discute a proveniência dos dados em sistemas multisistemas (SMA), mostrando uma nova perspectiva do atual estágio de uso dos sistemas de proveniência, ou seja, o que beneficia a proveniência dos dados e os problemas envolvidos na ausência de informações sobre as ações dos sistemas que fazem uso do SMA. Além disso, o artigo menciona algumas oportunidades de pesquisa que ainda estão inexploradas nos sistemas multiagentes.

Palavras-chave: Proveniência de Dados; Sistemas Multiagentes; Proveniência em SMA.

Responsável por publicações:

Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC-Rio Departamento de Informática
Rua Marquês de São Vicente, 225 - Gávea
22453-900 Rio de Janeiro RJ Brasil
Tel. +55 21 3114-1516 Fax: +55 21 3114-1530
E-mail: bib-di@inf.puc-rio.br

Sumário

1 Introdução

Sistemas Multiagentes (SMAs) são sociedades nas quais entidades heterogêneas e designadas individualmente (agentes) trabalham para realizar metas comuns ou independentes [19]. Com o aumento do uso de sistemas complexos, os agentes de software tornaram-se mais frequentemente incorporados em sistemas comerciais e industriais. A escolha de programar sistemas complexos usando linguagens orientadas a agentes é devido a algumas de suas características, como autonomia, pró-atividade e auto adaptação. Assim, o uso de agentes para a construção de sistemas complexos é considerado uma abordagem promissora em muitas áreas [20].

Os agentes de software podem automatizar uma série de ações, mas não podem fazer tudo. Conforme abordado por [1], um agente pode se comunicar com outros agentes ou com seres humanos, reagir a mudanças no ambiente e procurar atingir um objetivo específico. Um agente pode trabalhar sozinho ou interagindo com outros, criando assim um sistema multiagente. Os agentes podem ser classificados por dois tipos principais: (i) agentes que não apresentam inteligência - classificados apenas como agentes de software e (ii) agentes que apresentam algum grau de inteligência - classificados como agentes de software inteligentes [1].

Na maioria dos casos, os agentes são desenvolvidos usando o modelo BDI (Belief-Desire-Intention) [2]. O modelo BDI, conforme descrito por [21], são ações derivadas de um processo de raciocínio prático, que consiste em duas etapas. O primeiro passo faz a seleção de um conjunto de desejos que devem ser alcançados, de acordo com as crenças do agente. O segundo passo é responsável por determinar como esses desejos podem ser alcançados, como resultado do passo anterior. O BDI é composto de três atitudes mentais:

- Crenças - Elas representam a visão fundamental do agente em relação ao seu ambiente, outros agentes, interações com outros agentes e crenças sobre suas próprias crenças. O último pode levar a conflitos internos;
- Desejos - Eles representam o estado motivacional do agente, onde escolhe os desejos possíveis de acordo com algum critério, encorajando-o a realizar as tarefas para as quais foi projetado. Um desejo pode estar em conflito com as crenças do agente ou com outro desejo;
- Intenções - Intenções representam o plano de ação escolhido pelo agente, ou seja, se um agente decide seguir um objetivo específico, esse objetivo se torna uma intenção. As intenções são formadas a partir de um processo de deliberação, mas também podem conter as intenções iniciais inseridas pelo usuário.

Em [3] os autores propõem que os humanos estão envolvidos na maioria dos processos de software e as decisões tomadas explicam os efeitos dos processos. Com a inserção de agentes de software inteligentes na execução de tarefas, a autonomia do agente pode introduzir erros no processo e levar a falhas do sistema. Assim, para desenvolver um agente, é necessário um alto grau de atenção e conhecimento das ações do agente. Para cada decisão que o agente faz, existem aspectos positivos e negativos. Analisar o comportamento do agente não é uma tarefa trivial, pois pode adaptar e modificar seu comportamento. Com essas adaptações, podem surgir conflitos internos que não foram identificados durante o teste na construção do agente. Dado que, para uso em sistemas completos e exigindo um alto grau de precisão de ações, é necessário registrar todas as ações realizadas pelos agentes, bem como os dados manipulados por eles, para que qualquer

adaptação, conflito ou falha possa ser identificado. Uma maneira de registrar essas ações é por meio de dados de proveniência.

A Proveniência pode abordar mudanças no ambiente, a identificação de erros, o impacto que o erro causou e ainda pode causar, ou seja, pode seguir todas as ações que o sistema realizou. Para conhecer as ações dos sistemas autônomos, é necessário considerar as ações realizadas pelos agentes inteligentes, as ações humanas, os objetivos estipulados para os agentes e a informação sobre o meio onde os agentes atuam [4].

Em [5], os autores mostram que a proveniência dos dados deve ser modelada e armazenada para posterior análise e estar disponível para diferentes formas, como ontologias para máquinas de inferência, mineração de dados ou consultas específicas. Além disso, é importante determinar como os traços de proveniência serão capturados, porque a falta de registro de uma atividade pode invalidar a replicação das etapas, bem como a identificação da falha como um todo. Vale ressaltar que a captura da proveniência não é trivial [29], e trata principalmente de processos que envolvem agentes inteligentes e seres humanos.

Este artigo está estruturado da seguinte forma: a Seção II discute a proveniência em sistemas multiagentes e apresenta os campos de pesquisa. Ele também descreve os modelos de proveniência dos dados existentes e discute como a proveniência dos dados pode ser usada no SMAs. A Seção III mostra o estado da arte de algumas plataformas multiagente e sua aplicação à proveniência. Finalmente, a Seção IV apresenta nossa consideração e trabalho futuro.

2 Sistemas Multiagentes e Proveniência

A proveniência de dados é um tipo de meta-dados gerados por aplicativos ou processos computacionais que descrevem os produtos de dados usados ou gerados no processo, permitindo que os dados sejam desambiguados e reutilizados [8]. O gerenciamento de dados está crescendo em complexidade [8] à medida que novas aplicações emergem com recursos mal acoplados, envolvendo sistemas descentralizados, processamento de rede e fontes de informação abundantes.

Um mecanismo de captura de proveniência precisa acessar os detalhes relevantes de uma tarefa computacional, registrando seus passos, informações de execução e anotações de usuários [6]; Isso é importante, uma vez que a proveniência de dados é um rico conjunto de informações, divergindo dos sistemas de logs. Recolher a procedência de dados automaticamente nas aplicações de *Internet of Things* (IoT) e sistemas multiagentes são fundamentais para a compreensão de certas ações e resultados, dada a complexidade do domínio e os aplicativos desenvolvidos. Esta relação entre a proveniência em sistemas multiagentes e os modelos existentes de proveniência será discutida nas subseções 2.1, 2.2 e 2.3.

2.1 Sistemas Multiagentes e Falta de Informação

Os sistemas que utilizam agentes inteligentes têm qualidades específicas que exigem que os usuários justifiquem suas atividades, uma vez que as decisões tomadas por esse sistema devem ser consideradas confiáveis, especialmente se o sistema for amplamente utilizado [9]. Em aplicações autônomas, a confiabilidade dos dados processados por outros sistemas também deve ser confiável, pois acabam impactando o resultado geral de um processo. Além disso, existem vários pontos de entrada onde as pessoas que usam o sistema podem inserir dados.

A proveniência dos dados nos permite acompanhar as ações do sistema autônomo e sua interação com outros sistemas e humanos. Conforme apresentado por [9], a procedência dos dados levanta as questões:

1. Quem foi responsável pelo efeito X?
2. O efeito X corresponde ao que se destinava a acontecer?
3. Qual é o motivo (causal e intencional) do efeito X?

Além de responder a outras questões envolvendo sistemas multiagentes, tais como:

4. Quais foram as fontes de informação que alteraram o comportamento de um agente?
5. Quais agentes interagem uns com os outros?
6. Qual o ambiente em que o agente está inserido e o agente está ciente do meio ambiente?
7. Quais são as crenças, desejos e intenções do agente em cada momento?
8. O agente está tomando ações verdadeiramente autônomas?

Para responder a este tipo de pergunta, é necessário registrar informações sobre o agente e o sistema multiagente no qual ele está inserido, como mostrado na Figura 1. No entanto, as questões 4 a 8 vão além, envolvendo a captura do comportamento BDI do agente, como mostrado na Figura 2.

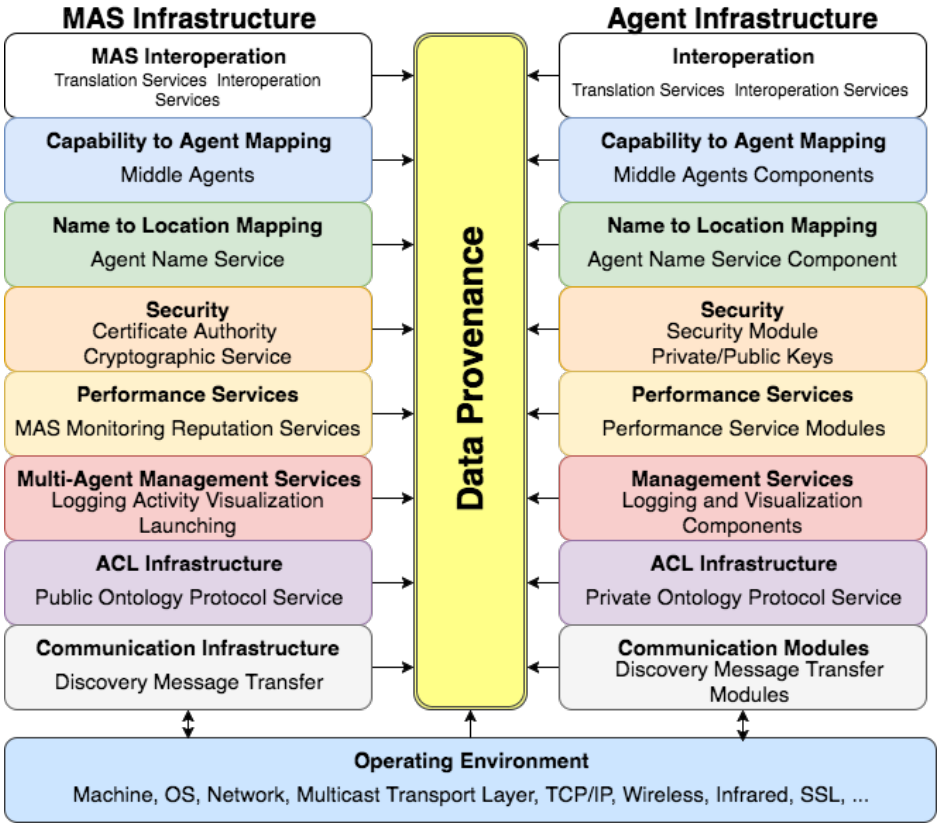

Fig. 1. Infraestrutura com a coleta de proveniência.

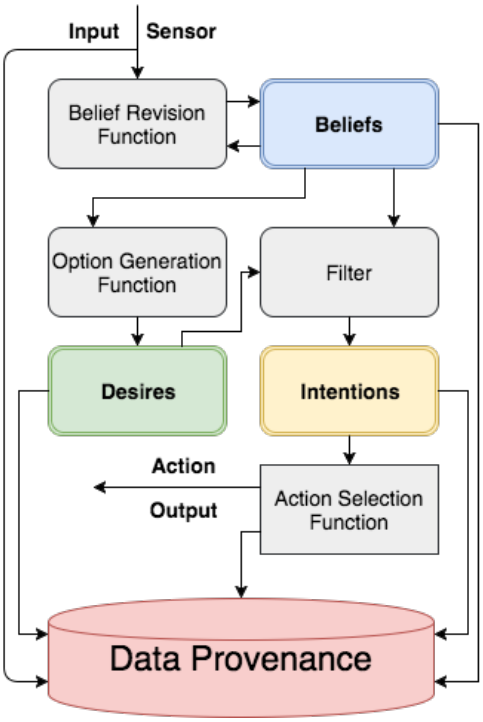

Fig. 2. Coleta de Proveniência no BDI.

Um sistema multiagente deve ser capaz de se auto adaptar, ou seja, estar atento às mudanças no ambiente e aos requisitos que definiram o SMA, para operar com novos requisitos. Conforme discutido em [9], todas as informações de proveniência em um sistema podem ser representadas como dados de proveniência trocados entre agentes, onde os agentes são objetos representados e suas relações causais como as interações entre objetos. Embora possamos documentar as relações causais na interação multiagentes, a relação agente e ser humano é mais complexa, porque os motivos que levam as pessoas a realizar certas ações não podem ser rastreados, conforme discutido por [10].

A proveniência de dados foi aplicada em vários sistemas que podem envolver o uso de sistemas multiagentes, entre os quais podemos destacar alguns campos de pesquisa. Um exemplo é E-Science, que é ciência assistida por computador, onde a proveniência de dados foi aplicada juntamente com sistemas multiagente para tratar dados em redes, conforme discutido por [11] [14] [15]. Na área da computação em nuvem, como abordado por [7] [16], a proveniência atua em uma camada superior, considerando diferentes propósitos, com foco em aplicações web.

No campo das aplicações computacionais de E-Health, a proveniência foi aplicada para monitorar indivíduos submetidos a tratamento e sistemas multiagentes, de modo que cada ação seja registrada e qualquer anomalia identificada, permitindo seguir cada etapa do processo [12]. A resposta de emergência e qualquer outro campo proveniente da proveniência, conforme discutido em [13], é para a simulação de ambientes ou para auxiliar a tomada de decisão em situações de risco.

A proveniência dos dados, juntamente com os sistemas multiagentes, tem sido aplicada em diferentes áreas de conhecimento, como mostrado. Embora [4] discuta a modelagem de dados para sistemas autônomos, não trata de modelos clássicos de procedência, como OPM [17] ou PROV [18], por exemplo. Como tal, cada sistema implementa a captura de dados dos indivíduos que utilizam o sistema desenvolvido e os sistemas multiagente de uma maneira que se adequa a eles.

A falta de um modelo padrão para sistemas multiagentes causa outros problemas, como a integridade, interoperabilidade, representatividade e confiabilidade dos dados. Ambos os modelos (OPM e PROV) serão apresentados e comparados abaixo, e depois aplicados em sistemas multiagentes.

2.2 Comparando os Modelos de Proveniência

O OPM [17] foi o primeiro modelo de proveniência amplamente utilizado e emergiu de uma conferência específica sobre o assunto, o Workshop Internacional de Proveniência e Anotações (IPAW). O modelo OPM representa o uso de dados digitais e está relacionado à documentação, derivação e anotação do mesmo. Mais tarde, um outro modelo de proveniência foi desenvolvido pelo grupo de proveniência do W3C e foi denominado PROV [18].

PROV não é uma expansão do OPM. É um novo modelo desenvolvido pelo W3C que está ganhando força para se tornar o modelo padrão em sistemas de captura de proveniência. O objetivo desta seção é fazer uma comparação entre esses dois modelos.

O modelo OPM é baseado em três vértices e suas relações causais são representadas como arcos de um gráfico. Os vértices principais do OPM são:

- Artefato: entidades imutáveis que podem representar um objeto físico ou uma representação digital de um sistema informático;
- Processo: uma ação ou conjunto de ações executadas ou causadas por artefatos que resultarão em novos artefatos;
- Agente: uma entidade contextual que atua como um catalisador para um processo, facilitando, controlando ou afetando sua execução.

As relações causais do OPM indicam relações de dependência e/ou causalidade. No total, existem cinco relações que classificam: (i) uso, (ii) geração, (iii) controle, (iv) desencadeamento e (v) derivação. Todos os vértices e relações causais nos permitem trabalhar com registros temporais e são específicos para cada propósito [6]. Cada uma das relações está especificada abaixo:

- used: Indica a relação de dependência de um artefato na execução de um processo;
- wasGeneratedBy: Indica que um processo foi responsável pela geração de um artefato;
- wasControlledBy: Indica que um processo foi controlado por um ou mais agentes;
- wasTriggeredBy: Indica que um processo foi iniciado por outro processo;
- wasDerivedFrom: Indica que um artefato se originou de outro artefato.

O modelo PROV consiste em uma família de doze documentos propostos pelo W3C. Esses documentos são divididos em modelo de dados, restrições, semântica, ontologia e anotações, conforme explicado por [18]. O PROV procura ser amplo, abordando a proveniência sob diferentes perspectivas e propósitos. Existem diferenças nos vértices em relação à OPM e novas relações causais. A proveniência pode ter três pontos principais: (i) nos agentes; (ii) em processos e, (iii) em objetos. Os vértices principais da PROV são:

- Entidade: as entidades que representam algo físico, digital, conceitual ou qualquer coisa com alguns aspectos fixos e, as entidades podem ser reais ou imaginárias;

- Atividade: promove atividades relacionadas a entidades e tem um período específico de tempo;
- Agente: Tem alguma responsabilidade por uma atividade, por uma entidade ou por atividades de outro agente. Um agente pode representar uma pessoa, uma organização ou um agente de software.

O modelo PROV possui dez relacionamentos que aumentam a representatividade da informação proveniente frente ao modelo OPM. As relações causais no modelo PROV são:

- used: representa um relacionamento onde uma atividade usou uma entidade;
- wasGeneratedBy: Representa uma entidade que foi gerada por uma atividade;
- wasAssociatedWith: Representa uma associação entre um agente e uma atividade;
- wasAttributedTo: Representa uma entidade que foi atribuída a um agente;
- actedOnBehalfOf: representa uma responsabilidade ou hierarquia entre os agentes;
- wasRevisionOf: Indica uma derivação, onde a entidade derivada é uma revisão de outra entidade existente, como uma correção;
- wasDerivedFrom: Indica uma derivação entre entidades com um caráter evolutivo ou adaptativo;
- wasInformedBy: Representa uma relação entre atividades, onde a atividade A informa que ela usou uma entidade gerada pela atividade B;
- wasStartedBy: Representa uma relação entre atividade e entidade, registrando o início da ação da forma temporal;
- wasEndedBy: Representa uma relação entre atividade e entidade, registrando o termo da ação da forma temporal.

A vantagem de usar o PROV em comparação com o OPM é a existência de relações específicas para agentes, o modelo PROV foca na responsabilidade dos dados e questões históricas, relações de agentes e entre os outros vértices. A Tabela 1 resume a comparação entre os modelos PROV e OPM.

Tabela 1. Comparação entre o OPM e o PROV.

OPM	PROV
Artifact	Entity
Process	Activity
Agent	Agent
Used	used
wasGeneratedBy	wasGeneratedBy
wasControlledBy	wasAssociatedWith
wasTriggeredBy	wasStartedBy
wasDerivedFrom	wasDerivedFrom
-	wasEndedBy
-	wasRevisionOf
-	wasAttributedTo
-	wasInformedBy
-	actedOnBehalfOf

2.3 Sistemas Multiagentes e as Implicações da Proveniência

Os autores em [29] propuseram o framework FProvW3C. Eles aplicaram essa estrutura para capturar dados de um sistema que usa agentes para classificação de gases expelidos por seres humanos. Este artigo apresenta alguns detalhes de como a proveniência dos dados pode ser aplicada para rastrear ações em sistemas multiagentes. Este trabalho pode ser definido como a base para responder as questões levantadas no método 5W2H, que permite, a qualquer momento, a identificação dos dados, processos e ações mais importantes dentro de um projeto, unidade de produção ou sistema.

O 5W2H é um método composto por sete questões que, além de identificar os dados, processos e ações, também permite identificar quem é responsável por uma ação, o que a pessoa responsável pela ação fez e por que ele fez isso. As sete questões da técnica 5W2H adaptada para o contexto dos sistemas multiagentes são:

- (What) O que o agente fez?
- (Who) Quem (ou qual agente) fez isso?
- (Where) Onde aconteceu (ação ou falha)?
- (When) Quando ocorreu a ação ou falha?
- (Why) Por que ocorreu a ação ou falha?
- (How) Como ocorreu a ação ou falha?
- (How much) Quanto isso custou ao sistema? (Esta última pergunta visa classificar o efeito de uma ação tomada, ou uma falha que ocorreu no sistema e o impacto).

Embora simples, a técnica 5W2H [31] permite a análise e o conhecimento sobre um determinado processo, problema ou ação a ser realizado ou que já ocorreu. Quando aplicado em sistemas multiagentes, ele ajuda com:

- Diagnóstico: a técnica 5W2H permite a investigação de um problema ou processo e a detecção de falhas e pontos de vulnerabilidade. Para lidar com esses aspectos, é necessário aumentar o nível de informação do sistema, caso em que a estrutura FProvW3C pode ser usada.
- Plano de ação: o teste de software em sistemas multiagentes não é uma tarefa trivial [32], uma vez que o agente pode ter comportamento autônomo mesmo quando segue as normas. A proveniência dos dados ajuda na construção de uma base de informações do agente, onde é possível (i) verificar seu plano de ação (suas decisões) sobre o que deve ser feito e (ii) eliminar comportamentos anormais que possam causar alguma anomalia no sistema.
- Documentação: o uso da proveniência possibilita documentar as etapas do sistema, extrair relatórios de execuções, falhas, pontos de vulnerabilidade e resultados produzidos por uma determinada entrada.

O método 5W2H nos permitiu dividir os dados em diferentes partes, o que, por sua vez, mostrou quais pessoas, ou agentes de software, estavam operando cada situação e cada ação. Também mostrou em qual momento aconteceu uma certa atividade, qual a sequência de ações que produziu o resultado, como a ação foi realizada e quais recursos foram gastos. Depois disso, alinhamos as questões apresentadas pelo método 5W2H com o uso da proveniência de dados em sistemas multiagentes. Como resultado, será possível capturar a dinâmica do sistema e responder as questões levantadas na seção 2.1.

3 Proveniência em SMA

Conforme discutido por [22], a alta heterogeneidade e a grande quantidade de plataformas de agentes disponíveis criaram um problema para os desenvolvedores, que é a escolha da plataforma certa ou mais adequada para um determinado problema.

Além disso, [22] salienta que a web está se movendo em direção a uma aldeia verdadeiramente global, conectando pessoas e conhecimento com o objetivo de construir uma rede de conteúdo armazenado na web. Como resultado, as máquinas podem entender o significado dos dados e satisfazer os pedidos das pessoas.

Vamos discutir como as principais plataformas de agentes tratam informações da proveniência dos dados e como isso pode contribuir para as ações dos agentes e ajudá-los a entender essas ações. Esta questão é importante porque um sistema que usa agentes de software deve ser bem pensado, planejado e avaliado, especialmente se as decisões envolvem seres humanos.

O *Java Agent Development Framework* (JADE) [23] é uma estrutura de desenvolvimento voltada para o progresso dos sistemas multiagentes de acordo com as especificações do FIPA. Embora o JADE tenha um *sniffer* para rastrear mensagens trocadas em um ambiente, o *sniffer* é aplicado somente quando os comportamentos do agente de depuração são analisados e as informações capturadas pelo *sniffer* não são salvas e não permitem consulta posterior - algo que poderia ser melhor utilizado.

O *Jack Intelligent Agent* (JACK) [24] é uma estrutura para o desenvolvimento de sistemas multiagentes, com um ambiente maduro e multiplataforma, voltado para a construção de sistemas multiagentes comerciais. O JACK usa o modelo BDI [2] e fornece suas próprias ferramentas de planejamento gráfico e de linguagem de planejamento baseadas em Java. O JACK, semelhante ao JADE, possui um ambiente de depuração de agente, que consiste em uma interface de exibição de interação de agentes e troca de mensagens entre agentes. O JACK não permite o armazenamento de logs de depuração ou a análise subsequente para encontrar falhas de execução ou rastrear o comportamento do agente.

O *JADEX* (JADE eXtension) [25] é uma camada de agente racional em cima do JADE, permitindo a criação de agentes de software inteligentes. Os agentes implementados com a JADEX seguem o bem conhecido modelo BDI na camada de design e implementação. De acordo com [25], a JADEX permite a geração de logs, que não estão estruturados e destinam-se a ajudar na depuração de aplicativos.

Semelhante a JADEX, temos BDI4JADE [26], que implementa a arquitetura BDI em JADE. Um dos principais objetivos da plataforma BDI4JADE é fornecer um ambiente para implementar diversas aplicações. A plataforma aproveita a infraestrutura JADE e amplia as relações que promovem a modularidade do agente, que é uma preocupação de engenharia de software relevante para o desenvolvimento de aplicações comerciais. BDI4JADE é baseado no JADE; Portanto, o *sniffer* tem as mesmas características.

JASON [27] é uma plataforma para o desenvolvimento de sistemas multiagentes com a linguagem AgentSpeak [28]. Este idioma é um dos idiomas abstratos mais influentes baseados na arquitetura BDI. Os sistemas desenvolvidos por meio da JASON utilizam o AgentSpeak em muitos casos para sistemas reativos. JASON é um intérprete completo para AgentSpeak, que também inclui comunicação entre agentes interativos baseados em fala. JASON possui uma ferramenta de depuração; no entanto, é limitado a informações simples sobre os agentes e as mensagens trocadas.

Entre as principais plataformas para a construção de sistemas multiagentes, podemos observar que, embora existam depuradores ou sistemas de log, não são projetados para rastrear ações de agentes, identificar falhas ou criar uma base para o comportamento do

agente em tempo de execução. Isso pode ser feito usando a proveniência de dados, modificando assim as arquiteturas das plataformas disponíveis. Por exemplo, com o modelo PROV, é possível armazenar os dados de forma estruturada, facilitando sua compreensão e análise.

4 Considerações Finais e Trabalhos Futuros

A proveniência dos dados é uma necessidade em muitos cenários, especialmente aqueles que possuem execuções complexas e são sensíveis à mudança, como em sistemas multiagentes. Tais recursos podem permitir: (i) ter um histórico de cada agente, com os passos que foram executados, bem como as informações recebidas e transmitidas, e (ii) nos permitem rastrear as auto adaptações que podem surgir. O modelo PROV é projetado para armazenar a proveniência dos dados de forma detalhada, focando as responsabilidades dos agentes em cada item de proveniência.

Este trabalho propõe a aplicação do modelo de proveniência do W3C PROV em sistemas multiagentes, permitindo a captura e armazenamento de dados de proveniência que podem ser usados para identificar erros no sistema e comportamentos auto adaptativos, além de rastrear as ações do agente para melhor compreender suas decisões. A proveniência dos dados nos permite rastrear a origem dos dados e os processos de derivação que ocorreram entre a fonte dos dados e o estado em que os dados são encontrados em um determinado momento. Se considerarmos que o modelo de proveniência contribui para avaliar a qualidade dos dados e consequentemente o processo que o gerou, isso nos permite aumentar a confiança nas ações tomadas pelos agentes dentro de um sistema multiagente.

Entre as diferentes plataformas de sistemas multiagentes, não há preocupações sobre os logs gerados ou sua utilização para identificar erros. Os dados capturados ou produzidos por agentes, juntamente com suas ações, na maioria dos casos são exibidos em terminais, sem representação adequada. Os dados de um sistema multiagente são valiosos na compreensão da dinâmica do sistema, e tanto a captura como o armazenamento são atividades críticas que devem ser bem planejadas e executadas para não invalidar ações.

A proveniência dos dados pode ser usada na construção de bases de conhecimento do SMA para ajudar na: (i) rastreabilidade das ações; (Ii) identificação de erros; (iii) acompanhamento das etapas de um sistema e, (iv) determinação da viabilidade de análise e verificação de resultados.

Conforme apresentado anteriormente, em sistemas multiagentes há muitas oportunidades para aplicar a proveniência dos dados. Para o trabalho futuro, pretendemos acoplar o modelo de proveniência do W3C PROV com as plataformas do sistema multiagente apresentadas. Ao usar a proveniência de dados para capturar dados em um MAS, será possível suportar: (i) semântica e sintaxe de dados, (ii) ontologia, (iii) mineração de dados e (iv) exportação de dados para outros formatos, como XML ou JSON.

Agradecimentos

Os autores agradecem a CAPES, Faperj, CNPq, PUC-Rio, FIVJ e LES pelo apoio e incentivo na pesquisa.

Referências

[1] Silva, José Carlos Tavares da. (2004). Um modelo para avaliação de aprendizagem no uso de ferramentas síncronas em ensino mediado pela Web (Doctoral dissertation, PUC-Rio). (In Portuguese)

[2] Georgeff, M., Pell, B., Pollack, M., Tambe, M., & Wooldridge, M. (1998, July). The belief-desire-intention model of agency. In International Workshop on Agent Theories, Architectures, and Languages (p. 1-10). Springer Berlin Heidelberg.

[3] Nunes, I., Chen, Y., Miles, S., Luck, M., & Lucena, C. (2012, June). Transparent provenance derivation for user decisions. In International Provenance and Annotation Workshop (p. 111-125). Springer Berlin Heidelberg.

[4] Miles, S., Munroe, S., Luck, M., & Moreau, L. (2007, May). Modelling the provenance of data in autonomous systems. In Proceedings of the 6th international joint conference on Autonomous agents and Multi-agent systems (p. 50). ACM.

[5] Dalpra, H. L., Costa, G. C., Sirqueira, T. F., Braga, R., Werner, C. M., Campos, F., & David, J. M. N. (2015). Using Ontology and Data Provenance to Improve Software Processes. In Proceedings of the Brazilian Seminar on Ontologies (p. 10-21).

[6] Freire, J., Koop, D., Santos, E., & Silva, C. T. (2008). Provenance for computational tasks: A survey. Computing in Science & Engineering, 10(3).

[7] Muniswamy-Reddy, K. K., Macko, P., & Seltzer, M. I. (2010, February). Provenance for the Cloud. In FAST (Vol. 10, p. 15-14).

[8] Simmhan, Y. L., Plale, B., & Gannon, D. (2005). A survey of data provenance techniques. Computer Science Department, Indiana University, Bloomington IN, 47405.

[9] Miles, S., Groth, P., Munroe, S., Luck, M., & Moreau, L. (2007, May). AgentPrIMe: Adapting MAS designs to build confidence. In International Workshop on Agent-Oriented Software Engineering (p. 31-43). Springer Berlin Heidelberg.

[10] Ariely, D. (2008). Predictably irrational (p. 20). New York: HarperCollins.

[11] Jami, S., & Shaikh, Z. (2008). A Multi-agent-based architecture for data provenance in semantic grid. In Proceedings of the International Multi-Conference of Engineers and Computer Scientists (IMECS). Hong Kong: Newswood Publications.

[12] Kifor, T., Varga, L. Z., Vazquez-Salceda, J., Alvarez, S., Willmott, S., Miles, S., & Moreau, L. (2006). Provenance in agent-mediated healthcare systems. IEEE Intelligent Systems, 21(6), p. 38-46.

[13] Naja, I., Moreau, L., & Rogers, A. (2010, June). Provenance of decisions in emergency response environments. In International Provenance and Annotation Workshop (p. 221-230). Springer Berlin Heidelberg.

[14] Szomszor, M., & Moreau, L. (2003, November). Recording and reasoning over data provenance in web and grid services. In OTM Confederated International Conferences "On the Move to Meaningful Internet Systems" (p. 603-620). Springer Berlin Heidelberg.

[15] Barker, A., & Mann, B. (2006, July). Agent-based scientific workflow composition. In Astronomical Data Analysis Software and Systems XV (Vol. 351, p. 485).

[16] Edwards, P., Pignotti, E., & Corsar, D. (2011). Provenance on the Web, Leaving the Walled Garden Behind. In: Proceedings of the ACM WebSci'11 (p. 1-2)

[17] Moreau, L., Clifford, B., Freire, J., Futrelle, J., Gil, Y., Groth, P., & Plale, B. (2011). The open provenance model core specification (v1. 1). Future generation computer systems, 27(6), p. 743-756.

[18] Missier, P., Belhajjame, K., & Cheney, J. (2013, March). The W3C PROV family of specifications for modelling provenance metadata. In Proceedings of the 16th International Conference on Extending Database Technology (p. 773-776). ACM.

[19] y López, F. L. (2003). Social Power and Norms (Doctoral dissertation, University of Southampton).

[20] Zambonelli, F., Jennings, N. R., & Wooldridge, M. (2003). Developing Multi-agent systems: The Gaia methodology. ACM Transactions on Software Engineering and Methodology (TOSEM), 12(3), 317-370.

[21] Viana, M. L.; Paulo Alencar; LUCENA, C. A Modeling Language for Adaptive Normative Agents. In: EUMAS, 2016, Valencia. European Conference on Multiagent Systems, 2016.

[22] Kravari, K., & Bassiliades, N. (2015). A survey of agent platforms. Journal of Artificial Societies and Social Simulation, 18(1), 11.

[23] Bellifemine, F. L., Caire, G., & Greenwood, D. (2007). Developing multi-agent systems with JADE (Vol. 7). John Wiley & Sons.

[24] Howden, N., Rönnquist, R., Hodgson, A., & Lucas, A. (2001, May). JACK intelligent agents-summary of an agent infrastructure. In 5th International conference on autonomous agents.

[25] Pokahr, A., Braubach, L., & Lamersdorf, W. (2005). Jadex: A BDI reasoning engine. Multi-agent programming, 149-174.

[26] Nunes, I., Lucena, C. J. P. D., & Luck, M. (2011). BDI4JADE: a BDI layer on top of JADE. ProMAS 2011, 88-103.

[27] Bordini, R. H., & Hübner, J. F. (2005, June). BDI agent programming in AgentSpeak using Jason. In International Workshop on Computational Logic in Multi-Agent Systems (pp. 143-164). Springer Berlin Heidelberg.

[28] Rao, A. (1996). AgentSpeak (L): BDI agents speak out in a logical computable language. Agents breaking away, 42-55.

[29] Sirqueira, T., Viana, M., Nascimento, N & Lucena, C. (2017). A Software Framework for Data Provenance. In *SEKE: Software Engineering and Knowledge Engineering* (pp. 615-619).

[30] Valentim, M. (2008). Métodos e técnicas de planejamento. UNESP. (in portuguese)

[31] Pimentel, J., Santos, E., Castro, J., & Franch, X. (2012). Anticipating requirements changes-using futurology in requirements elicitation. International Journal of Information System Modeling and Design (IJISMD), 3(2), 89-111.

[32] Cunha, F., da Costa, A. D., Viana, M., & de Lucena, C. J. P. (2015, December). JAT4BDI: An Aspect-Based Approach for Testing BDI Agents. In Web Intelligence and Intelligent Agent Technology (WI-IAT), 2015 IEEE/WIC/ACM International Conference on (Vol. 2, pp. 186-189). IEEE.