Pontifícia Universidade Católica
do Rio de Janeiro

Tassio Ferenzini Martins Sirqueira

Sistemas Autônomos Explicáveis por meio de Proveniência de Dados

Tese de Doutorado

Tese apresentada como requisito parcial para obtenção do grau de Doutor pelo Programa de Pós–graduação em Informática do Departamento de Informática da PUC-Rio.

Orientador: Prof. Carlos José Pereira de Lucena

Rio de Janeiro
Dezembro de 2019

Pontifícia Universidade Católica
do Rio de Janeiro

Tassio Ferenzini Martins Sirqueira

Sistemas Autônomos Explicáveis por meio de Proveniência de Dados

Tese apresentada como requisito parcial para obtenção do grau de Doutor pelo Programa de Pós–graduação em Informática da PUC-Rio. Aprovada pela Comissão Examinadora abaixo.

Prof. Carlos José Pereira de Lucena
Orientador
Departamento de Informática – PUC-Rio

Prof. Alessandro Fabrício Garcia
Pontifícia Universidade Católica do Rio de Janeiro – PUC-Rio

Prof. Marco Antônio Casanova
Pontifícia Universidade Católica do Rio de Janeiro – PUC-Rio

Prof. Marco Antônio Pereira Araújo
Universidade Federal de Juiz de Fora – UFJF

Profª. Regina Maria Maciel Braga Villela
Universidade Federal de Juiz de Fora – UFJF

Rio de Janeiro, 05 de Dezembro de 2019

Tassio Ferenzini Martins Sirqueira

Formado Mestre em Ciência da Computação pela Universidade Federal de Juiz de Fora (2016), Bacharel em Sistemas de Informação pelo Centro de Ensino Superior de Juiz de Fora (2013) e Técnico em Informática Industrial pelo SENAI-JFN (2008). Atualmente é professor no Centro de Ensino Superior de Juiz de Fora, pesquisador no Laboratório de Engenharia de Software (LES) na PUC-Rio e avaliador de cursos superiores do INEP/MEC. Tem experiência na área de Computação com ênfase em Engenharia de Software e Banco de Dados.

Ficha Catalográfica

Sirqueira, Tassio Ferenzini Martins

Sistemas Autônomos Explicáveis por meio de Proveniência de Dados / Tassio Ferenzini Martins Sirqueira; orientador: Carlos José Pereira de Lucena. – Rio de janeiro: PUC-Rio, Departamento de Informática, 2019.

v., 109 f: il. color. ; 30 cm

Tese (doutorado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática.

Inclui bibliografia

1. Proveniência de Dados – Teses. 2. Sistemas Multiagentes – Teses. 3. Proveniência de Dados;. 4. Sistema Multiagente;. 5. FProvW3C;. 6. Prov-BDI4JADE;. 7. Sistemas Autônomos Explicáveis;. 8. Provenência em SMA.. I. Lucena, Carlos José Pereira de. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Dedico este trabalho a minha esposa Jéssica, minha mãe Ivana, minha avó Apparecida, meu avô Rubens (em memória), ao Moacir e a Beatriz e a todos os meus professores, por suas orientações e ajudas, pois sem vocês este trabalho e muitos dos meus sonhos não se realizariam.

Agradecimentos

Em primeiro lugar gostaria de agradecer a Deus por ter me dado forças para chegar até aqui e superar todas as barreiras.

Agradecer o todo o apoio e incentivo da minha espose Jéssica, minha mãe Ivana, minha avó Apparecida, ao Moacir e a Beatriz, sempre me motivando em seguir em frente.

A meu orientador, prof. Lucena, por todas as reuniões e momentos de discussão, me dando liberdade e proporcionando amadurecer minhas ideias.

A Vera, por todas as nossas conversas e os melhores cafés que já tomei. Agradeço também por seu seu esforço em me ajudar e em especial sua amizade.

Ao Custódio e ao Ingo, que abriram a porta de sua casa e me cederam um local para residir durante meus estudos.

As meus amigos do LES, em especial com o Marx, o Francisco, a Nathália e a Chrystinne, com quem pude conviver e trocar muitas ideias, tonando esse tempo mais agradável.

Aos professores da PUC-Rio pelas imensas contribuições ao meu aprendizado.

A todos os meus colegas professores do Centro de Ensino Superior de Juiz de Fora e do Instituto Vianna Júnior, e da MRS, pelos incentivos e apoio na continuação de meus estudos.

Aos meus ex-orientadores que me incentivaram a seguir em frente.

Aos membros da Banca Examinadora pelo trabalho de avaliação.

E por fim, não menos importantes, aos meus amigos que sempre estiveram comigo ajudando e apoiando minhas conquistas.

O presente trabalho foi realizado com o apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001

Resumo

Sirqueira, Tassio Ferenzini Martins; Lucena, Carlos José Pereira de. **Sistemas Autônomos Explicáveis por meio de Proveniência de Dados**. Rio de Janeiro, 2019. 109p. Tese de Doutorado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Determinar a proveniência dos dados, isto é, o processo que levou a esses dados, é vital em muitas áreas, especialmente quando é essencial que os resultados ou ações sejam confiáveis. Com o crescente número de aplicações baseadas em inteligência artificial, criou-se a necessidade de torná-las capazes de explicar seu comportamento e responder às suas decisões. Isso é um desafio, especialmente se as aplicações forem distribuídas e compostas de vários agentes autônomos, formando um Sistema Multiagente (SMA). Uma maneira fundamental de tornar tais sistemas explicáveis é rastrear o comportamento do agente, isto é, registrar a origem de suas ações e raciocínios, como em uma "depuração onisciente". Embora a ideia de proveniência já tenha sido explorada em alguns contextos, ela não foi extensivamente explorada no contexto de SMA, deixando muitas questões para serem compreendidas e abordadas. Nosso objetivo neste trabalho é justificar a importância da proveniência dos dados para SMA, discutindo quais perguntas podem ser respondidas em relação ao comportamento do SMA, utilizando a proveniência e ilustrando, através de cenários de aplicação, os benefícios que a proveniência proporciona para responder a essas questões. Este estudo envolve a criação de um *framework* de software, chamado FProvW3C, que suporta a coleta e armazenamento da proveniência dos dados produzidos pelo SMA, que foi integrado a plataforma BDI4JADE (41), formando o que denominamos de Prov-BDI4JADE. Por meio desta plataforma, utilizando exemplos de sistemas autônomos, demostramos com rigor que, o uso da proveniência de dados em SMA é uma solução sólida, para tornar transparente o processo de raciocínio e ação do agente.

Palavras-chave

Proveniência de Dados; Sistema Multiagente; FProvW3C; Prov-BDI4JADE; Sistemas Autônomos Explicáveis; Provenência em SMA.

Abstract

Sirqueira, Tassio Ferenzini Martins; Lucena, Carlos José Pereira de (Advisor). **Autonomous Systems Explainable Through Data Provenance**. Rio de Janeiro, 2019. 109p. Tese de doutorado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Determining the data provenance, that is, the process that led to those data, is vital in many areas, especially when it is essential that the results or actions be reliable. With the increasing number of applications based on artificial intelligence, the need has been created to make them capable of explaining their behavior and be responsive to their decisions. This is a challenge especially if the applications are distributed, and composed of multiple autonomous agents, forming a Multiagent System (MAS). A key way of making such systems explicable is to track the agent's behavior, that is, to record the source of their actions and reasoning, as in an "omniscient debugging". Although the idea of provenance has already been explored in some contexts, it has not been extensively explored in the context of MAS, leaving many questions to be understood and addressed. Our objective in this work is to justify the importance of the data provenance to MAS, discussing which questions can be answered regarding the behavior of MAS using the provenance and illustrating, through application scenarios, to demonstrate the benefits that provenance provides to reply to these questions. This study involves the creation of a software framework, called FProvW3C, which supports the collects and stores the provenance of the data produced by the MAS, which was integrated with the platform BDI4JADE (41), forming what we call Prov-BDI4JADE. Through this platform, using examples of autonomous systems, we have rigorously demonstrated that the use of data provenance in MAS is a solid solution to make the agent's reasoning and action process transparent.

Keywords

Data Provenance; Multiagent System; FProvW3C; Prov-BDI4JADE; Explainable Artificial Intelligence; Provenance in MAS software framework.

Sumário

Lista de figuras

Lista de tabelas

Lista de Abreviaturas

BDI – Crenças-Desejos-Intenções (*Beliefs-Desires-Intentions*)
CBT – Código Brasileiro de Trânsito
CES/JF – Centro de Ensino Superior de Juiz de Fora
CRUDs – Criar-Ler-Atualizar-Deletar (*Create-Read-Update-Delete*)
ESE – Engenharia de Software Experimental
GQM – *Goal-Question-Metrics*
IA – Inteligência Artificial
IoT – Internet das Coisas (*Internet of Things*)
KW – *Kruskal-Wallis* ML – Aprendizado de Máquina (*Machine Learning*)
MWW – *Mann-Whitney-Wilcoxon*
NoSQL – Não apenas SQL (*Not Only SQL*)
OO – Orientação a Objetos
OPM – Modelo Aberto de Proveniência (*Open Provenance Model*)
POA – Programação Orientada a Aspectos
POO – Programação Orientada a Objetos
SGBD – Sistema Gerenciador de banco de Dados
SMA – Sistema Multiagente
W3C – *World Wide Web Consortium*
XAI – Inteligência Artificial Explicável (*Explainable Artificial Intelligence*)

Nós só podemos ver um pouco do futuro, mas o suficiente para perceber que há muito o que fazer.

Alan Turing, *Matemático e Cientista da Computação.*

1
Introdução

Sistemas computacionais já fazem parte do nosso cotidiano, e para o desenvolvimento de novos sistemas para atender uma demanda cada vez mais pujante, por softwares eficazes e eficientes, dependemos de soluções que possam ser escaláveis, que possam lidar com o crescimento explosivo dos dados, e que sejam adaptadas para à internet e também para o chamado "Internet das Coisas" (IoT) (1).

Esses novos sistemas podem demandar o uso de inteligência artificial (IA), representada por sistemas multiagentes, alinhada com técnicas de aprendizado de máquina (ML), demonstram sucesso prático em muitos domínios de aplicação, como por exemplo, em sistemas autônomos de direção, reconhecimento de fala ou de recomendação, conforme (19). Com uma série de atividades ocorrendo de forma autônoma, o mundo passa a manipular grandes quantidades de dados, que devem ser atualizados em tempo real, e que de modo geral, são utilizados por milhares de pessoas, dispositivos ou outros sistemas autônomos.

A introdução de componentes autônomos aos novos sistemas, muitas vezes passam a ser um diferencial primordial ao novo cenário em que vivemos (4) e a quebra do paradigma de desenvolvimento de sistemas simplesmente CRUDs (*Create, Read, Update, Delete*), emerge o uso de inteligência artificial, aprendizado de máquina, agentes de software ou mesmo de sistemas multiagentes, que podem ser entendidos como inteligência artificial distribuída, onde um agente deixa de ser um simples objeto do ponto de vista da engenharia de software (ES) (2, 3).

Conforme definido por (6) "Um agente é um sistema de computador que está situado em algum ambiente e que é capaz de ação autônoma nesse ambiente para atingir seus objetivos de projeto". Ao longo dos últimos anos, o uso de agentes de software e sistemas multiagente tem crescido por conta da demanda dos novos sistemas (5), com necessidades de automação e da inserção de IA e ML.

A inteligência artificial é um grande campo da computação que engloba lógica, probabilidade, matemática, entre outras áreas, para que as máquinas sejam capazes de resolverem problemas que até então eram reservados para os

humanos.

Com isso, nos últimos anos, a chamada Inteligência Artificial Explicável (XAI) tem recebido significativa atenção (22, 19, 21, 23), devido à necessidade desses sistemas serem capazes de explicar seu comportamento e serem responsáveis por suas decisões e ações, principalmente quando tais decisões têm um impacto significativo na vida dos seres humanos.

Esse trabalho visa o uso de proveniência de dados em sistemas autônomos, onde o termo proveniência de dados é definido por (97) como um registro da história da derivação dos dados, que possibilita a reprodutibilidade, interpretação dos resultados e diagnostico de problemas.

Nas próximas seções deste capítulo serão abordadas uma contextualização sobre o porque é importante rastrear as ações de sistemas autônomos, definiremos os problemas a serem atacados, as questões de pesquisa que visamos e os objetivos com esta pesquisa, além da abordagem que será utilizada e a forma com que o trabalho está organizado.

1.1 Contextualização

Com um aumento no desenvolvimento de sistemas complexos, fornecendo algum grau de inteligência, os agentes de software tornaram-se mais frequentemente incorporados em plataformas comerciais e industriais (3). A escolha de desenvolver sistemas complexos usando a tecnologia de agente de software, assim como as linguagens, plataformas e técnicas associadas, se deve ao fornecimento de abordagens reutilizáveis, prontas para implementar recursos desafiadores, como proatividade, raciocínio, aprendizado e comunicação distribuída.

Esses sistemas autônomos podem funcionar isolados ou serem uma coleção de sistemas autônomos heterogêneos, que colaboram para atingir um objetivo em específico ou objetivos comuns ao grupo. Os sistemas autônomos que são compostos de várias entidades individuais heterogêneas, são geralmente chamados de Sistemas Multiagentes (SMA) (20).

O problema central do uso de sistema multiagentes, é que eles são considerados uma caixa preta e mesmo que entendamos os princípios matemáticos subjacentes dos agentes, falta uma representação declarativa explícita do conhecimento, portanto, temos dificuldade em gerar estruturas explicativas subjacentes, como metadados. Os metadados são marcos ou pontos de referência que permitem delimitar a informação, podendo resumir ou referenciar o conteúdo de uma fonte.

Um meio-chave de tornar esses sistemas multiagentes explicáveis é rastrear o comportamento dos agentes, isto é, registrar de forma clara suas ações

e raciocínios. Embora a ideia de proveniência de dados tenha sido explorada em alguns contextos, principalmente *e-science*, ela tem sido pouco explorada no contexto de sistemas autônomos, deixando muitas questões em aberto que devem ser compreendidas e abordadas, conforme mostraremos a seguir.

Para contextualizar a importância da proveniência dos dados em sistemas autônomos, vamos utilizar como exemplo, as regras de intercessão de trânsito extraídas do Código Brasileiro de Trânsito (CBT), onde temos um cruzamento sem semáforo, e por exemplo, um conjunto de carros autônomos.

Esses carros devem interagir entre si e definir quem tem a prioridade neste cruzamento, caso todos cheguem ao mesmo tempo nele. Nesse contexto, os agentes de software são construídos sobre normas, devendo respeitar o CBT, entretanto, existe um conflito entre os artigos 29 e 38 do CBT, conforme apresentado por (61), deixando a resolução deste conflito a cargo dos motoristas.

O conflito consiste em determinar quem tem a prioridade. Neste caso, o uso da proveniência poderia explicar as interações que ocorreram entre cada agente, seus raciocínios internos, e permitir rastrear as ações que cada agente realizou para determinar suas ações. Este cenário, pode ser visto na Figura 1.1, extraída de (44).

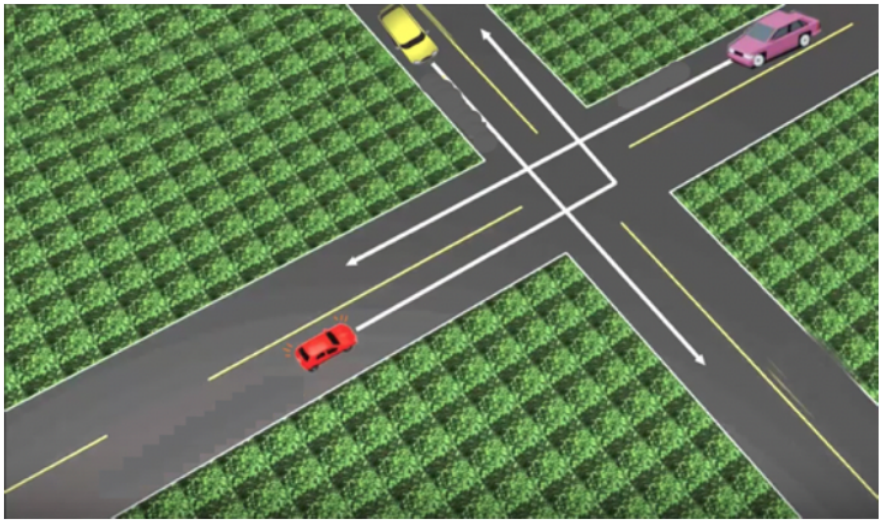


Figura 1.1: Cenário de interseção de trânsito no Brasil

Além deste contexto, outros como os apresentados em (56, 57, 58, 59, 60), podem se beneficiar com a proveniência de dados, como mecanismo de rastrear e explicar as ações executadas em sistemas autônomos.

Dado os recentes acontecimentos envolvendo os carros autônomos da Uber (25), da Tesla (62) e o avião da Boing (63), podemos observar a importância de rastrear sistemas autônomos, e tornar possível no futuro evitar que casos desse tipo voltem a ocorrer, utilizando a proveniência de dados.

Com um modelo de proveniência sólido e alinhado as tarefas computacionais que deve abranger, torna-se possível definir e mensurar os resultados, seja por meio de métodos quantitativos, qualitativos ou de classificação semântica.

1.2 Definição do Problema

Um agente é uma entidade autônoma que busca realizar alguma tarefa à qual foi designado. Entretanto, desenvolver software orientado a agente com apenas um agente não é o suficiente, pois uma das características fundamentais destes é a comunicação.

Já um sistema multiagente (SMA) pode ser entendido como um conjunto de agentes autônomos, que buscam a solução de um problema que está além de suas capacidades individuais (7) e conforme (8) explica, sistemas multiagentes é uma abordagem promissora para a construção de sistemas complexos, contudo, manter a qualidade e confiabilidade do software é uma atividade árdua, devido a complexidade natural do software e pelas interações que ocorrem entre os agentes.

Sistemas multiagentes podem ser projetados sobre estruturas homogêneas, heterogêneas ou híbrida (8), onde a arquitetura do sistema multiagente deve permitir que os agentes interajam, para garantir a funcionalidade do sistema. A arquitetura de um agente determina sua estrutura e modo de operação, que podem ser classificados com base na aplicabilidade, na forma de interação e no grau de "inteligência" do agente, onde temos as classificações dispostas na Tabela 1.1 de (9).

Na classificação de (9) podemos concluir que a arquitetura deliberativa segue um modelo simbólico do mundo, guiado por crenças, desejos e intenções, chamado de modelo BDI (Belief-Desire-Intention) (10). Na arquitetura reativa não existe raciocínio, apenas respostas a estímulos externos e já a arquitetura híbrida, busca um equilíbrio entre as duas anteriores.

Além da classificação por arquitetura, 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 (11). Essas classificações buscam definir padrões para implementação usando sistemas multiagentes.

A arquitetura do software provê uma visão global do sistema, o que permite: i) compreender o funcionamento do sistema; ii) especificar pontos críticos; iii) identificar possibilidade de reuso; iv) verificar e validar o sistema e; v) definir critérios de qualidade que permitam o sistema evoluir posteriormente.

Tabela 1.1: Classificação das arquiteturas de agentes

Arquitetura de Agentes	Variação do nome	Descrição
Arquitetura Deliberativa	Arquitetura BDI Arquitetura baseada em lógica Arquitetura baseada em metas	Possui um modelo simbólico do mundo. As decisões são tomadas via raciocínio lógico. Possui um conjunto de metas e intenções, onde é elaborado um plano baseando-se nessas metas. Possui certas restrições sobre a construção do modelo simbólico. Agentes complexos.
Arquitetura Reativa	Arquitetura reflexiva	Sem modelo simbólico interno. As decisões tomadas são implementadas em alguma forma de mapeamento direto da situação para a ação, usando regras de condição/ação.
Arquiteturas híbridas	Arquiteturas em camadas	As decisões tomadas via várias camadas de software. Mistura componentes das arquiteturas Deliberativa e Reativa.

Todo software é construído sobre alguma abstração e em softwares que em sua arquitetura utiliza um sistema multiagente, a abstração pode ser o próprio agente (12).

A escolha de programar sistemas autônomos complexos usando agentes de software é devido suas características, como autonomia, pró-atividade e auto adaptação (88). Contudo, para avaliar um sistema é necessário reunir informações suficientes de modo que as pessoas possam entender as execuções, abstraindo os detalhes irrelevantes e compreender as interações que ocorreram.

Os sistemas que utilizam agentes autônomos 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 (13). Em aplicações autônomas, a confiabilidade dos dados processados por outros sistemas também deve ser confiável, pois acaba impactando o resultado geral de um processo. Além disso, as aplicações podem possuir pontos de entrada, onde as pessoas que usam o sistema podem inserir dados.

Entretanto, se considerarmos sistemas autônomos complexos, com uso de agentes inteligentes, cobrir todas as ações dos agentes se tornam inviáveis. Conforme (14) demostram em seu trabalho, analisar a árvore de planos de objetivos, e que verificar se uma árvore de plano de metas tem um cronograma de execução, com relação aos requisitos de recursos, é NP-completo. Logo, não existe ainda uma solução para verificar em tempo polinomial as ações do agente presente na base do BDI, sendo assim, soluções como as de (24, 52, 53), tornam-se a mais viáveis neste caso, sendo pontual e parcial em relação a todas as ações que podem ocorrer dentro de um SMA.

Isso levará a intuição de que "agentes autônomos de software são difíceis de testar" (15), tanto com relação aos possíveis comportamentos dos agentes BDI, quanto à sua probabilidade de falhar. Essa definição derrogam a visão global das propostas supramencionadas, e impede que as pessoas possam compreender certas ações tomadas pelos sistemas multiagentes, subtraindo a confiabilidade de suas ações e levando a questões como as apresentadas por (16), que são:

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 outras questões envolvendo a interação dentro dos sistemas multiagentes e a seguridade de suas ações, 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 se o agente está ciente deste meio?
7. Quais são as crenças, desejos e intenções do agente em cada momento (de falha ou sucesso)?
8. O agente está tomando ações verdadeiramente autônomas?

Para responder a estes tipos de perguntas, é necessário registrar informações sobre o comportamento do agente e do sistema multiagente no qual está inserido, e para isso, este trabalho apresenta como uma abordagem viável, o uso de proveniência de dados.

Com o uso da proveniência é possível registrar as informações em cada situação atípica (como uma falha), podemos identificar quais dados foram recebidos e as ações tomadas pelo sistema autônomo antes da situação atípica ocorrer.

Nos casos apresentados na seção 1.1, a informação de proveniência poderia ajudar a explicar os acontecimentos. Nesses casos, analisando os dados coletados pelos sensores e o "raciocínio" dos veículos antes do acidente, seria possível identificar de onde surgiu a falha que culminou no acidente. Em cenários críticos, como o de veículos autônomos, a proveniência dos dados e o histórico gerado se tornam essenciais.

No próximo capítulo teremos uma visão geral sobre sistemas de *Logs* e de modelos de proveniência de dados, e suas aplicabilidades, além de iniciarmos uma discussão sobre as vantagens dos modelos de proveniência em relação aos de *Logs*.

1.3 Questão de Pesquisa

Hoje é cada vez mais importante determinar a origem dos dados, ajudando a interpretá-los. Ou seja, conhecer o processo de derivação que gerou um conjunto de dados, auxilia a compreender como ocorreu o processo e seus desdobramentos, por meio da sua derivação histórica. Portanto, armazenar os dados, metadados e seu processo de derivação pode ser uma importante ferramenta para auxiliar em uma análise, garantindo sua autenticidade, legitimidade e confiabilidade.

Para auxiliar na formulação das hipóteses de pesquisa, foi utilizado o método GQM (*Goal, Question, Metric*) (64). A Tabela 1.2 define o objetivo específico do estudo, baseado na abordagem GQM conforme (65). O GQM é um enfoque da mensuração orientada em metas, que ajuda na definição e implementação de uma questão de pesquisa (66) .

Com base no GQM, definiu-se as hipóteses de pesquisa que englobam de forma geral o trabalho e de modo específico o desenvolvimento da solução, a saber, o *framework* de software, chamado FProvW3C (26) e a plataforma de construção de sistemas autônomos inteligentes (BDI4JADE (41)), com uso do *framework* desenvolvido, denominada Prov-BDI4JADE , que serão apresentados no capítulo 4.

Tabela 1.2: GQM do estudo

Analisar:	O comportamento de sistemas autônomos (IA)
Com o propósito de:	Explicar as ações geradas (XAI)
Em respeito ao:	Agente de software empregado
Do ponto de vista:	Desenvolvedores e Pesquisadores
No contexto:	Sistemas Multiagentes (MAS)

As hipóteses (H0) são apresentadas a seguir com suas respectivas alternativas (H1):

Hipótese Geral:

- H0: O uso de proveniência de dados auxilia na análise das ações realizadas por sistemas autônomos, tornando-os explicativos.
- H1: O uso de proveniência de dados não auxilia na análise das ações realizadas por sistemas autônomos, tornando-os explicativos.

Hipótese Específica:
Framework FProvW3C

- H0: O uso do *framework* de software FProvW3C, para coleta de dados de proveniência em sistemas autônomos, possibilita torná-los explicativos.
- H1: O uso do *framework* de software FProvW3C, para coleta de dados de proveniência em sistemas autônomos, não possibilita torná-los explicativos.

Prov-BDI4JADE

- H0: O uso da plataforma Prov-BDI4JADE possibilita abstrair os detalhes da configuração de coleta dos dados de proveniência, tornando o sistemas autônomos explicativos, sem demandar conhecimento sobre os modelos de proveniência.
- H1: O uso da plataforma Prov-BDI4JADE não possibilita abstrair os detalhes da configuração de coleta dos dados de proveniência, o que não torna os sistemas autônomos explicativos, demandando conhecimento sobre os modelos de proveniência.

Com base nas hipóteses mencionadas anteriormente, pretende-se concluir que: *A coleta e o armazenamento dos dados de proveniência para posterior análise, utilizando um modelo de proveniência de dados, é capaz de fornecer as informações necessárias, sobre sistemas autônomos, de modo a facilitar a análise das decisões tomadas pelos agentes de software.*

1.4 Objetivos da Pesquisa

Uma das formas mais comuns de registrar informações do sistema para compreender sua execução é através do uso de *logs*. A geração de *logs* é a geração de registros de eventos, que proveem informações sobre a utilização do software à medida que acontecem e preservam esses registros para análise posterior. Entretanto, alguns pontos sobre o uso dos tradicionais sistemas de *logs* como o Log4J (18) ainda estão em aberto, como, por exemplo, o que deve ser registrado, a frequência desse registro, o que deve conter e etc.

Considerando esses aspectos, o objetivo deste trabalho é demostrar que os sistemas de *logs* como o Log4J, que é utilizado em plataformas de sistemas autônomos, como a BDI4JADE (41), não são suficientes para coletarem dados que demostram de forma clara e precisa os acontecimentos do sistema autônomo.

Conforme apresentado anteriormente, como é impossível verificar em tempo hábil as assertivas em um sistema multiagente, por meio de testes de software, para compreender as ações do sistema, identificar falhas ou adaptações do sistema e tornar esse processo inteligível, necessitamos recorrer a meios melhores do que os sistemas de *logs*, onde nesse trabalho buscamos o uso de proveniência de dados, tornando os sistemas autônomos explicáveis, respondendo as questões da seção anterior.

Com base nisso, nosso objetivo geral é definir e implementar um *framework* para coletar dados de proveniência, que permita armazená-los e posteriormente, analisar o comportamento de um sistema autônomo.

Já os nossos objetivos específicos são:

- Pesquisar as abordagens existentes na literatura, que tratam do uso de proveniência de dados em SMA e da XAI;
- Analisar os modelos para a captura e armazenamento dos dados de proveniência em SMA;
- Desenvolver um *framework* para coletar e armazenar os dados de proveniência do SMA, seguindo a especificação básica do PROV;
- Avaliar o *framework* desenvolvido, acoplando-o a plataforma BDI4JADE, para construção de sistemas autônomos explicáveis por meio da proveniência.

Na próxima serão apresentaremos a abordagem proposta e as motivações que levaram à mesma.

1.5 Abordagem Proposta

Conforme (23), existe um desafio histórico da IA, que é: i) Determinar quais são as representações apropriadas de conhecimentos que demonstram alguma veracidade com o domínio que está sendo capturado; e ii) Quais mecanismos de raciocínio oferecem a base para transmitir uma inferência computada em termos desse modelo.

A proveniência de dados é um tipo de metadado gerado por aplicações ou processos computacionais e descrevem os produtos de dados utilizados ou gerados no processo, permitindo que os dados sejam desambiguados e reutilizados (28). O gerenciamento de dados está crescendo em complexidade conforme novos aplicativos surgem com recursos fracamente acoplados, envolvendo sistemas descentralizados, processamento em rede e fontes abundantes de informação.

Dadas essas características, um mecanismo para capturar dados de proveniência precisa acessar detalhes relevantes de uma tarefa computacional, registrando seus passos, informações de execução e anotações do usuário (29). Isso é importante porque a proveniência de dados é um rico conjunto de informações, divergindo dos registros de *logs* do sistema. A coleta automática de dados em aplicações e sistemas multiagentes é fundamental para o entendimento de determinadas ações e resultados, dada a complexidade do domínio e das aplicações desenvolvidas. Essa relação entre proveniência em sistemas multiagentes e modelos existentes de proveniência será discutida no capítulo seguinte.

A proveniência dos dados deve ser modelada e armazenada para análise posterior e estar disponível para diferentes usos, como ontologias para máquinas de inferência, mineração de dados ou consultas específicas (27). Além disso, é importante determinar como os rastros de proveniência serão capturados, porque a falha em registrar uma atividade pode invalidar a replicação das etapas de uma execução do sistema, bem como a identificação da falha como um todo.

A coleta dos dados de proveniência devem ser tratados como um interesse transversal dentro do sistema autônomo, com isso, o uso de mecanismos separados para cuidar desse "aspecto" se faz necessário. Conforme (67), a programação orientada a aspectos (POA), enfatiza o uso efetivo de ferramentas e mecanismos de linguagens particulares a fim de concretizar as propriedades e os conceitos relacionados a aspectos no nível da implementação, normalmente no escopo da programação orientada a objetos (POO). Ainda conforme (67), a programação orientada a aspectos é mais complexa do que uma simples

combinação de aspectos ortogonais com classes em pontos de combinação bem definidos, e (68) complementa que, o desenvolvimento de softwares orientados a aspectos é um paradigma para modularizar *concerns* cujos objetos não são capazes de capturar explicitamente.

Entretanto, vale ressaltar que *crosscutting* entre aspectos e componentes é o principal problema da POA. O uso da POA pode afetar classes e estruturas de aplicações, e é necessário uma semântica de composição consistente para evitar conflitos ou interferir de forma irregular.

Este trabalho defende a importância do uso da proveniência de dados para tornar explicáveis SMA, discutindo quais questões podem ser respondidas em relação ao comportamento do SMA usando a proveniência de dados e a POA e, visa como continuidade demonstrar os benefícios que a proveniência de dados fornecem para responder as questões supramencionadas. Este estudo envolve o uso de um *framework*, a saber, o FProvW3C, que estende a possibilidade de coletar e armazenar a proveniência dos dados, produzidos por um SMA com uso de POA.

1.6 Organização do Trabalho

Este trabalho está organizado em cinco capítulos além desta introdução, conforme Figura 1.2. O capítulo 2 apresenta a fundamentação teórica do trabalho, abordando conceitos de sistemas multiagentes, sistemas de *logs* tradicionais, os principais modelos de proveniência de dados. Já no capítulo 3, apresentamos uma revisão *quasi* sistemática da literatura e os principais trabalhos que se relacionam com esta proposta.

No capítulo 4, apresenta-se uma visão geral do trabalho desenvolvido, o *framework* de software FProvW3C e sua integração com a plataforma de agentes BDI4JADE, formando o que denominamos de Prov-BDI4JADE, explicando sua estrutura, forma de coleta, armazenamento e disponibilização dos dados, de modo a fomentar a XAI.

O capítulo 5 trás uma avaliação do *framework* e do Prov-BDI4JADE, onde explicamos os experimentos realizados e os resultados obtidos, respondendo as questões de pesquisa. Por fim, no capítulo 6, são apresentadas algumas considerações finais, as contribuições obtidas, bem como algumas limitações e trabalhos futuros.

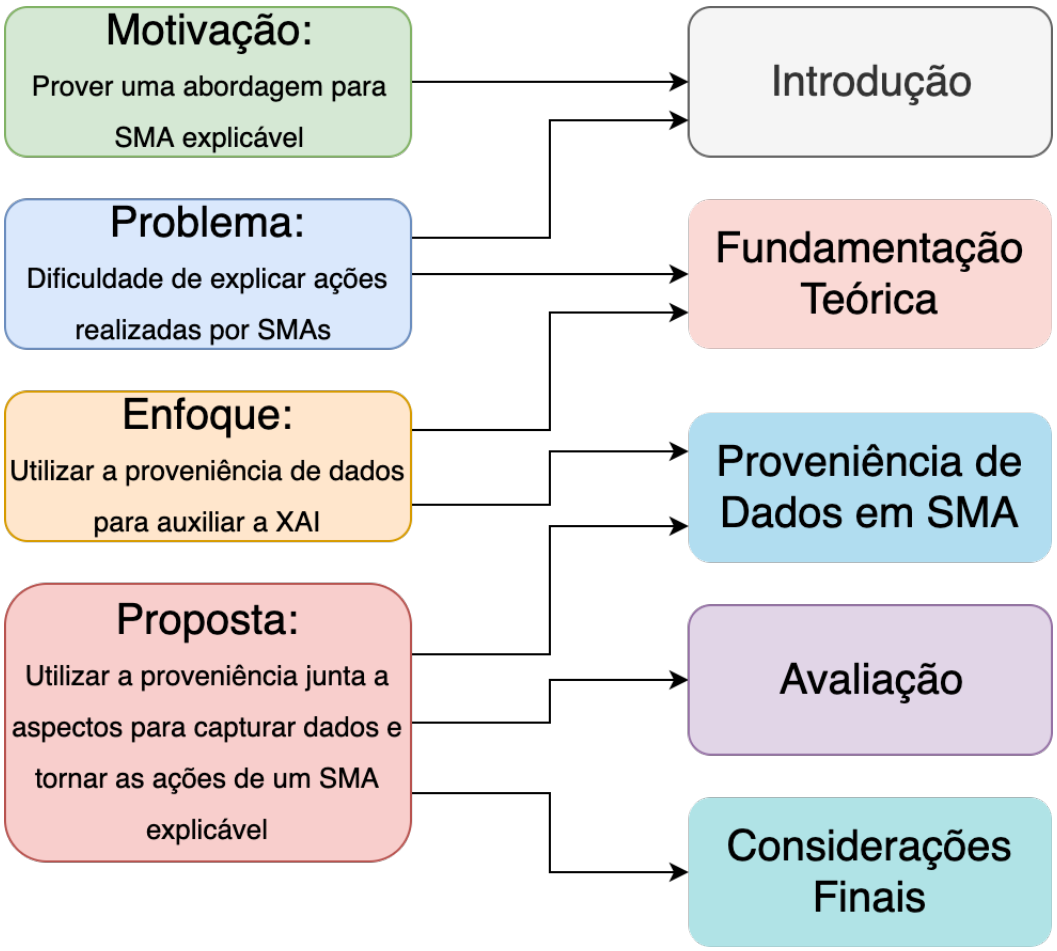


Figura 1.2: Organização do trabalho.

2
Fundamentação Teórica

A inteligência artificial não é um conceito novo em ciência da computação e conforme (30, 31) apud (19), experimentou muitos altos e baixos desde sua introdução formal como uma disciplina acadêmica. O grande objetivo da IA é fornecer os fundamentos necessários para desenvolvermos softwares que possam aprender autonomamente a partir da experiência anterior, de forma automática e sem intervenção humana, para realizar algum tipo de tarefa (32), um sistema multiagente (SMA) pode ser entendido como sinônimo de inteligência artificial distribuída.

Com isso, entender e tornar explicáveis as ações de um sistema autônomo é um desafio que ainda permanece em aberto (22, 21, 23). No geral, a eficácia total de todo o sucesso de um SMA é limitada pela sua capacidade do algoritmo de explicar seus resultados à humanos.

Um SMA é muito mais que uma simples inteligência artificial distribuída, e por isso é complexo entender certas ações que podem ocorrer em aplicações complexas. Na próxima seção teremos uma visão geral sobre SMA e por que hoje suas ações não são claras o suficiente para permitirem que a sua inteligência artificial seja explicável.

2.1
Sistemas Multiagentes

Conforme abordado por (11), um agente de software pode se comunicar com outros agentes ou com seres humanos, reagir a mudanças no ambiente e procura atingir um objetivo específico. Conforme já explicado, os agentes podem ser classificados como agentes de software e agentes de software inteligentes. Não é objetivo deste trabalho discutir o que é inteligência em um sistema multiagente.

Na maioria dos casos, os agentes inteligentes são desenvolvidos usando o modelo BDI (Belief-Desire-Intention) (10). No modelo BDI, conforme Figura 2.1 (extraída de (41)), as ações são 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 da etapa anterior.

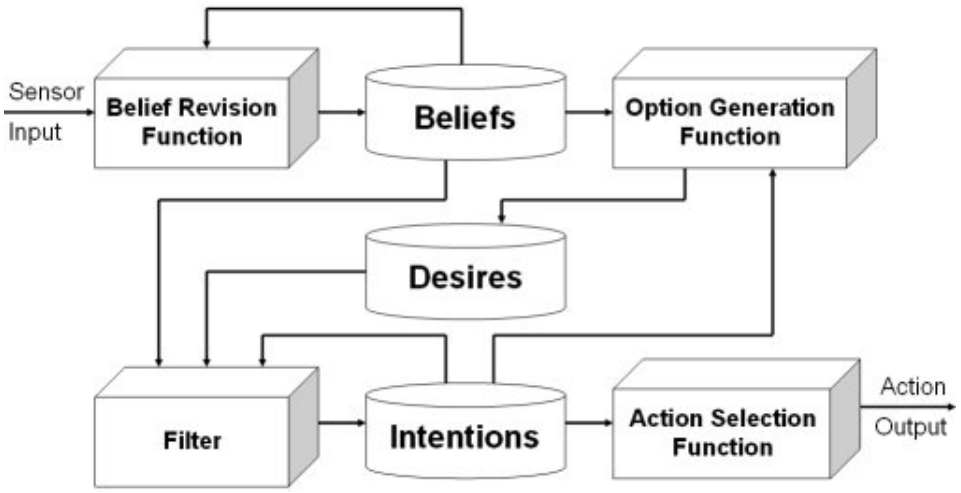


Figura 2.1: Processo de raciocínio BDI

O BDI é composto de três atitudes mentais:

– Crenças - 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. O uso do termo crença em vez de conhecimento reconhece que o que um agente acredita pode não ser necessariamente verdadeiro (e de fato pode mudar no futuro);

– Desejos - 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 outros desejos;

• O bjetivos : É um desejo que foi adotado para o agente realizar. O uso dos objetivos acrescenta a restrição adicional de que o conjunto de desejos ativos deve ser consistente. Por exemplo, não se deve ter objetivos simultâneos de ir a uma festa e ficar em casa - mesmo que ambos possam ser desejáveis.

– 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. Nos sistemas implementados, isso significa que o agente começou a executar um plano.

- P lanos : os planos são sequências de ações (receitas ou áreas de conhecimento) que um agente pode executar para atingir uma ou mais de suas intenções. Os planos podem incluir outros: meu plano de dar uma volta pode incluir um plano para encontrar as chaves do meu carro. Isso reflete que, no modelo de Bratman (89), os planos são inicialmente concebidos apenas parcialmente, com detalhes sendo preenchidos à medida que progridem.

Além do BDI, temos conforme explicado por (44), a arquitetura NBDI (45), que estende a arquitetura BDI (10) ao incluir funções relacionadas a normas para apoiar o raciocínio. Além disso, as normas são consideradas um conceito primário que influenciam a decisão do agente, quando raciocinando sobre as suas crenças, desejos e intenções.

Um agente de software deve ser conforme (33) : i) autônomo: agir sem a necessidade de intervenção humana; ii) reativo: perceber e responder a mudanças no ambiente; iii) pró-ativo: executar suas ações sempre que possível; iv) social: interagir com outros agentes, seja para cumprir com seus objetivos ou com os objetivos de outros agentes e; v) intencional: possuir crenças, desejos e objetivos e a intenção de cumpri-los.

Já um software que faz uso de um sistema multiagente pode apresentar uma arquitetura simples; composta por um único agente, o que vai contra os princípios de um sistema multiagente, onde à cooperação entre si permite resolver problemas além de sua capacidade individual, como supracitado; arquitetura moderada, formada por agentes homogêneos e distribuídos e; arquitetura complexa, composta por agentes heterogêneos, distribuídos e que apresentam algum grau de inteligência (12).

Em arquiteturas com mais de um agente, a comunicação entre os mesmos podem ocorrer conforme (12) como um quadro-negro, compartilhando um repositório de perguntas e respostas para a interação entre os agentes; comunicação por troca de mensagens, o que ocorre de maneira assíncrona, mas dependendo que o agente conheça seus vizinhos para se comunicar e a comunicação federativa; onde os agentes são divididos em grupos e a comunicação ocorre entre os grupos por meio de agentes facilitados ou controladores, de modo a reduzir o excesso de comunicação.

Para este último, as coordenações entre os agentes podem ocorrer como mestre-escravo ou mecanismo de mercado. Na mestre-escravo, os mestres são os agentes responsáveis por delegar as ações aos outros agentes e os escravos são os agentes responsáveis por executa-las. Já no mecanismo de mercado todos os agentes estão no mesmo nível e conhecem as ações que cada um pode executar, desta forma os agentes negociam as execuções das tarefas entre si, diretamente.

Essas classificações de arquiteturas de agentes, buscam definir padrões entre as arquiteturas que são implementadas usando sistemas multiagentes. Padrões de software podem se referir a diferentes níveis de abstração no desenvolvimento de sistemas, visto que existem padrões arquiteturais, padrões de análise, padrões de projeto, padrões de código, entre outros (34). Entretanto, se os usos de padrões não forem bem projetados podem criar lacunas no desenvolvimento do software, e essas lacunas dificultarem entender certas ações que estão ocorrendo em um software, por exemplo. O que torna compreender o sistema autônomo mais complexo.

2.1.1 Plataformas de Agentes

Existe uma grande quantidade de plataformas de agentes disponíveis (35), com diferentes características e com falta de critérios padronizados concretos, tornando a escolha da plataforma de agentes dependente da finalidade do sistema a ser desenvolvido. Portanto, escolher a plataforma certa ou adequada para um determinado problema ainda é uma dúvida para os desenvolvedores. Nesse contexto vamos apresentar algumas das principais plataformas de sistemas multiagentes e suas características.

A JADE[1] (36) é um *framework* totalmente implementado em Java e segue as especificações FIPA[2] (37). A plataforma do agente pode ser distribuída em máquinas heterogêneas e a configuração pode ser controlada através de uma interface remota. A configuração pode ser alterada em tempo de execução, movendo agentes de uma máquina para outra, conforme necessário. É um software livre, de código aberto e estável, distribuído pela *Telecom Itália*, detentora dos direitos autorais.

O sistema do JADEX[3] BDI (38), segue o modelo BDI e permite a programação de agentes de software inteligentes, em XML e Java. O projeto é conduzido pelo *Grupo de Sistemas Distribuídos e Sistemas de Informação da Universidade de Hamburgo* e licenciado sobre a GNU LGPL (69). Sua arquitetura baseia-se na noção de componentes ativos, que são conceitualmente baseados em SCA (arquitetura de componentes de serviço). Isso permite a decomposição hierárquica de componentes que interagem através de serviços, e se destina a funcionar em sistemas distribuídos, concorrentes e dinâmicos. O BDI é uma abstração que permite aos desenvolvedores gerenciar a complexidade do problema, como supracitado.

[1]Disponível em: <http://jade.tilab.com/>
[2]Disponível em: <http://www.fipa.org/>
[3]Disponível em: <https://www.activecomponents.org/>

Já o JACK[4] (39) é um ambiente multiplataforma, para construir, executar e integrar sistemas multiagentes. É construído sobre a lógica BDI e está inteiramente escrito em Java, mas ainda não suporta as especificações do FIPA. As especificações do FIPA (37) representam uma coleção de padrões que visam promover a interoperação de agentes heterogêneos e os serviços que eles podem representar. Já o Jason[5] (40) é desenvolvido em JAVA, sendo um interpretador do *AgentSpeak* (70), uma linguagem de programação lógica baseada em agente do BDI. Um dos pontos forte são os recursos customizáveis pelo usuário.

O BDI4JADE[6] (41) é uma extensão da plataforma JADE, construída em JAVA e que implementa a arquitetura BDI de agentes inteligentes. Como no JADE, cada agente tem sua própria execução, composta de um conjunto de intenções e capacidades, as crenças e planos são partes da capacidade de um agente. As mensagens são enviadas e recebidas como no JADE e os eventos podem ser relacionados às crenças ou objetivos. O BDI4JADE possui seu código fonte e documentação disponíveis online, com exemplos de uso e que está em constante evolução. A plataforma segue as especificações FIPA.

Conforme apresentado, existem diversas plataformas para diferentes objetivos, e implementadas de diferentes maneiras. Contudo, faltam padrões claros e abrangentes para a especificação, construção e manutenção de softwares com uso de sistemas multiagentes. Os SMA precisam ser bem projetados por causa de sua ampla quantidade de características, para se moldar a vários tipos de ambientes para atuação. Como já apresentado, os softwares devem refletir o mundo real, demandando que mudanças futuras não causem impacto na arquitetura do sistema. Para isso é necessário que o sistema multiagente tenha um foco de atuação bem definido para que os agentes possam ter objetivos concisos.

A seguir serão discutidos estes pontos e como a captura de informações do sistema multiagente podem auxiliar na compreensão das ações dos agentes, na identificação de falhas, no acompanhamento do sistema e como essas informações devem ser capturadas sem afetar a arquitetura do sistema multiagente dentro da arquitetura do software.

2.2 *Logs*

De modo geral, para avaliar um sistema é necessário reunir informações suficientes de modo que as pessoas possam entender as execuções, abstraindo os detalhes irrelevantes e compreender às interações que ocorreram. Com base

[4]Disponível em: <http://aosgrp.com/products/jack/>
[5]Disponível em: <http://jason.sourceforge.net/>
[6]Disponível em: <http://www.inf.ufrgs.br/prosoft/bdi4jade/>

nas informações provenientes do sistema em tempo de execução, podemos identificar ações não observadas em testes, que podem possibilitar melhorar a arquitetura, prolongar o seu decaimento e tornar as ações do sistema explicáveis.

Os registros de *logs* constituem uma fonte básica de informação, seja para a detecção de problemas, uso indevido do sistema, problemas com serviços ou hardwares, ou mesmo para rastrear (auditar) as ações executadas por um usuário ou um sistema multiagente. Os *logs* proveem informações sobre a utilização do software à medida que as ações acontecem e preservam esses registros para posterior análise.

Entretanto, alguns pontos sobre o uso de *logs* ainda estão em abertos, como por exemplo o que deve ser registrado, a frequência desse registro, o que deve conter e etc. Os *logs* se caracterizam por serem flexíveis, onde mediante configurações é possível registrar desde eventos críticos até praticamente todos os eventos de um sistema, mas em muitos casos não são modelados no projeto do sistema e cabe ao desenvolvedor determinar sobre o uso dos mesmos.

Como já dito, registrar informações de um software em sistemas complexos pode trazer benefícios, e uma das formas mais comuns é o uso de *logs*, utilizados pela maioria das plataformas de agentes apresentadas. Por meio dos arquivos de *logs* é possível registrar ações dos usuários, do sistema ou de agentes de software, visto que os *logs* registram quem acessou os recursos computacionais, aplicativos, arquivos de dados e utilitários, quando foi feito o acesso e que tipo de operações foram efetuadas, desde que programados para registrar isso pelo desenvolvedor do sistema autônomo.

Uma ferramenta bastante conhecida para geração de *logs* é a Log4J[7] (18). Ela divide os *logs* em 8 níveis hierárquicos, sendo eles: i) DEBUG; ii) INFO; iii) WARN; iv) ERROR; v) FATAL; vi) TRACE; vii) ALL e; viii) OFF. A grande maioria dos sistemas de *logs* seguem essa mesma característica, entretanto o gerenciamento de eventos envolve um conjunto mais amplo de atividades, que são: i) armazenamento; ii) compactação; iii) indexação e busca; iv) análise léxica e sobre o tipo de evento; v) visualização e em alguns casos; vi)a transmissão e recebimento, visto que os *logs* podem ser armazenados de forma centralizada ou distribuída, sendo elas online ou off-line, como discutida por (17).

Conforme (17), dependendo do software o conjunto de *logs* pode ser extenso e distribuído em várias máquinas, podendo muitas vezes trazer dados insuficientes sobre o contexto do sistema que não está representando os eventos por completo. Ainda conforme (17), considerando todos os diferentes contextos,

[7]Disponível em: <https://logging.apache.org/log4j/>

é difícil correlacionar os eventos criando links lógicos que poderiam explicar o seu comportamento, sendo necessário mais informações sobre o comportamento do sistema para entende-lo e os sistemas de *logs*, normalmente trazem apenas palavras-chaves em seus registros, ou explicações superficiais.

Como abordado por (27), o aumento de dados gerados torna a análise mais complexa e exige o uso de técnicas para permitir a análise adequada desses dados, extraindo registros que, de fato, contribuam para a melhoria do processo, e uma maneira de analisar esses dados é usar técnicas e modelos de proveniência.

A proveniência fornece um olhar além das especificações dos domínios e sugere a adoção de modelos disciplinados, onde a informação de proveniência de dados pode ser usada para aprender ou compreender métodos e regras de design (26). Atualmente, existem dois padrões principais para a captura de dados de proveniência: i) o modelo OPM (42), e ii) o modelo PROV (43), que serão abordados na sequência.

2.3 Proveniência de Dados

Embora (16) discutam a modelagem de dados para sistemas autônomos, não tratam os modelos clássicos de proveniência, como OPM (42) ou PROV (43), deixando que cada sistema possa implementar como lhe convém. A falta de um modelo padrão causa alguns problemas, como a falta de integridade, interoperabilidade, representatividade e confiabilidade dos dados.

O objetivo de se utilizar um modelo de proveniência, deve-se a possibilidade de troca de informações entre sistemas, utilizando uma camada de compatibilidade independente da tecnologia, apoiando a modelagem de qualquer "coisa" em um sistema computacional, possibilitando gerar regras de inferência sobre um conjunto de dados. A seguir serão apresentados os modelos OPM e PROV, ambos modelos de proveniência amplamente difundidos.

2.3.1 OPM

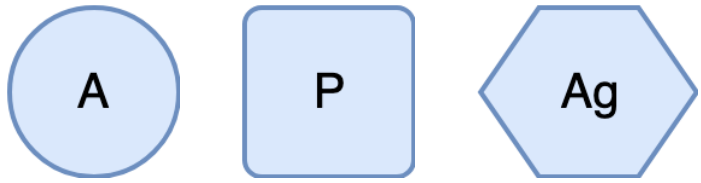
O OPM (42) foi o primeiro modelo de proveniência amplamente utilizado e emergiu de uma conferência específica sobre o assunto, o *Workshop Internacional de Proveniência e Anotações* (IPAW). O modelo OPM representa o uso de dados digitais e está relacionado à documentação, derivação e anotação do mesmo.

Conforme (71), o modelo OPM supõe-se que a proveniência pode ser representada por um grafo de causalidade anotada, que é um grafo acíclico di-

rigido, enriquecido com anotações capturadas de outras fontes de informações.

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 computacional. É representado conforme Figura 2.2(a);
- Processo: uma ação ou conjunto de ações executadas ou causadas por artefatos que resultarão em novos artefatos. É representado conforme Figura 2.2(b);
- Agente: uma entidade contextual que atua como um incentivador para um processo, facilitando, controlando ou afetando sua execução. É representado conforme Figura 2.2(c).

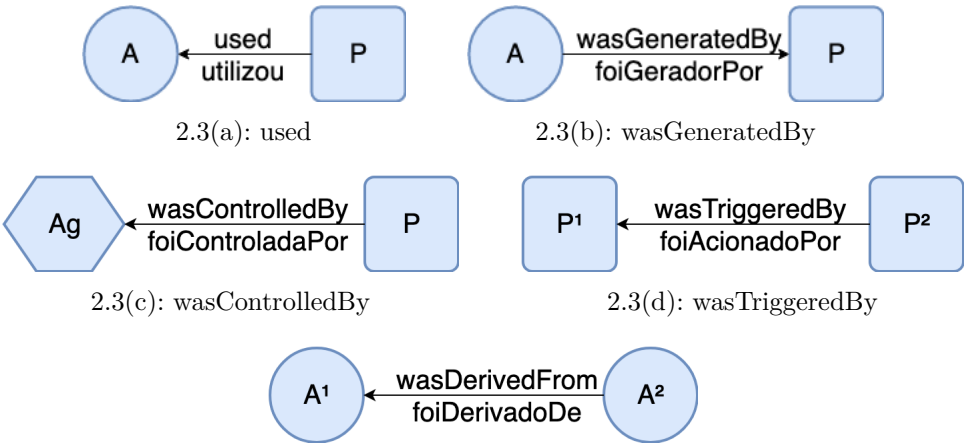


2.2(a): Artefato 2.2(b): Processo 2.2(c): Agente

Figura 2.2: Representação dos vértices do OPM

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 (29). Cada uma das relações está especificada abaixo:

- used: Indica a relação de dependência de um artefato na execução de um processo, representado conforme Figura 2.3(a);
- wasGeneratedBy: Indica que um processo foi responsável pela geração de um artefato, representado conforme Figura 2.3(b);
- wasControlledBy: Indica que um processo foi controlado por um ou mais agentes, representado conforme Figura 2.3(c);
- wasTriggeredBy: Indica que um processo foi iniciado por outro processo, representado conforme Figura 2.3(d);
- wasDerivedFrom: Indica que um artefato se originou de outro artefato, representado conforme Figura 2.3(e).



2.3(a): used

2.3(b): wasGeneratedBy

2.3(c): wasControlledBy

2.3(d): wasTriggeredBy

2.3(e): wasDerivedFrom

Figura 2.3: Representação das relações causais do OPM.

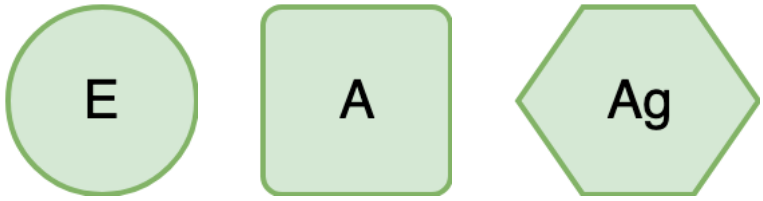
Mais tarde, um outro modelo de proveniência foi desenvolvido pelo grupo de proveniência do W3C e foi denominado PROV (43). PROV não é uma expansão do OPM. É um novo modelo desenvolvido pelo W3C, que está ganhando força para se tornar o modelo padrão em sistemas de proveniência.

2.3.2 PROV

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 (43). 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, sendo representadas pela Figura 2.4(a);
- Atividade: promove atividades relacionadas a entidades e tem um período específico de tempo, sendo representadas pela Figura 2.4(b);
- 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, sendo representado pela Figura 2.4(c).



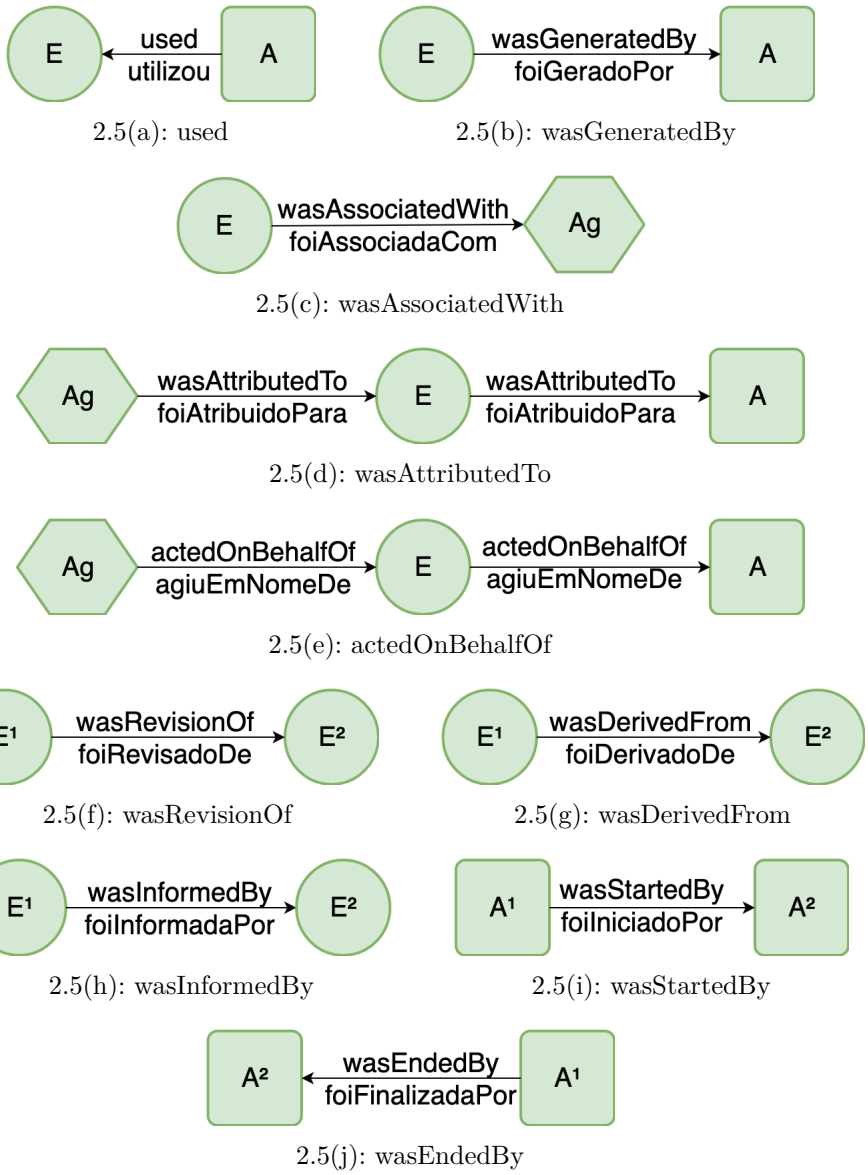
2.4(a): Entidade 2.4(b): Atividade 2.4(c): Agente

Figura 2.4: Representação dos vértices do OPM

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, conforme Figura 2.5(a);
- wasGeneratedBy: Representa uma entidade que foi gerada por uma atividade, conforme Figura 2.5(b);
- wasAssociatedWith: Representa uma associação entre um agente e uma atividade, conforme Figura 2.5(c);
- wasAttributedTo: Representa uma entidade que foi atribuída a um agente, conforme Figura 2.5(d);
- actedOnBehalfOf: representa uma responsabilidade ou hierarquia entre os agentes, conforme Figura 2.5(e);
- wasRevisionOf: Indica uma derivação, onde a entidade derivada é uma revisão de outra entidade existente, como uma correção, conforme Figura 2.5(f);
- wasDerivedFrom: Indica uma derivação entre entidades com um caráter evolutivo ou adaptativo, conforme Figura 2.5(g);
- wasInformedBy: Representa uma relação entre atividades, onde a atividade A informa que ela usou uma entidade gerada pela atividade B, conforme Figura 2.5(h);
- wasStartedBy: Representa uma relação entre atividade e entidade, registrando o início da ação da forma temporal, conforme Figura 2.5(i);
- wasEndedBy: Representa uma relação entre atividade e entidade, registrando o termo da ação da forma temporal, conforme Figura 2.5(j).

Além dessas relações apresentadas, existem outras relações causais, que podem ser divididas em dois subconjuntos: i) relações primárias e ii) secundárias (opcionais). A Figura 2.6 mostra as relações primárias e a Figura 2.7 apresenta as relações secundárias, ambas do PROV-DM (90).



2.5(a): used

2.5(b): wasGeneratedBy

2.5(c): wasAssociatedWith

2.5(d): wasAttributedTo

2.5(e): actedOnBehalfOf

2.5(f): wasRevisionOf

2.5(g): wasDerivedFrom

2.5(h): wasInformedBy

2.5(i): wasStartedBy

2.5(j): wasEndedBy

Figura 2.5: Representação das relações causais do PROV.

		Object		
		Entity	Activity	Agent
Subject	Entity	**WasDerivedFrom** Revision Quotation PrimarySource AlternateOf SpecializationOf HadMember	**WasGeneratedBy** WasInvalidatedBy	*R* *T* *L*	**WasAttributedTo**
	Activity	**Used** WasStartedBy WasEndedBy	*R* *T* *L*	**WasInformedBy**	**WasAssociatedWith**	*R*
	Agent	—	—	**ActedOnBehalfOf**

Figura 2.6: Relações Primárias do PROV

		Secondary Object		
		Entity	Activity	Agent
Subject	Entity	—	WasDerivedFrom (activity)	—
	Activity	WasAssociatedWith (plan)	WasStartedBy (starter) WasEndedBy (ender)	—
	Agent	—	ActedOnBehalfOf (activity)	—

Figura 2.7: Relações Secundárias do PROV

Cada representação do PROV é definido por um conjunto de restrições (PROV-CONSTRAINTS) e semânticas (PROV-SEM), que definem o modelo de dados (PROV-DM) e uma ontologia (PROV-O) para o mapeamento do modelo de dados. As regras definidas por essas especificações foram utilizadas para a construção da base do *Framework* FProvC3C, e são determinados conforme a Tabela 2.1.

Além das regras básicas do PROV (apresentadas na Tabela 2.1), durante o desenvolvimento do *Framework* FProvW3C, foi definida outra relação causal aplicada à SMA, que é oriunda da relação formada entre Agente, Entidade e Atividade. Dessa forma, toda relação de início, fim ou uso que liga uma Atividade a uma Entidade e, que por sua vez está atribuída a um Agente, foi definido que existe a relação causal *WasAssociatedWith*, conforme apresenta a Figura 2.8.

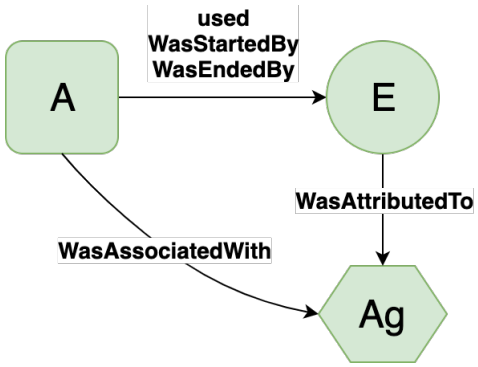


Figura 2.8: Relação causal formada pela tripla Agente, Entidade e Atividade.

Tabela 2.1: Principais regras do PROV aplicadas no FProvW3C.

PROV	Regra
Entity	IF entity(E,atrib) THEN 'entity' ∈ typeOf(E) & IF entity(c,[prov:type='prov:EmptyCollection']) THEN 'entity' ∈ typeOf(c) AND 'prov:Collection' ∈ typeOf(c) AND 'prov:EmptyCollection' ∈ typeOf(c)
Activity	IF activity(A,t1,t2,atrib) THEN 'activity' ∈ typeOf(A)
Agent	IF agent(Ag,atrib) THEN 'agent' ∈ typeOf(Ag)
used	IF used(id; A,E,t,atrib) THEN 'activity' ∈ typeOf(A) AND 'entity' ∈ typeOf(E)
wasGeneratedBy	IF wasGeneratedBy(id; E,A,t,atrib) THEN 'entity' ∈ typeOf(E) AND 'activity' ∈ typeOf(A)
wasAssociatedWith	IF wasAttributedTo(id; E,Ag,atrib) THEN 'entity' ∈ typeOf(E) AND 'agent' ∈ typeOf(Ag)
wasStartedBy	IF wasStartedBy(id; A2,E,A1,t,atrib) THEN 'activity' ∈ typeOf(A2) AND 'entity' ∈ typeOf(E) AND 'activity' ∈ typeOf(A1)
wasEndedBy	IF wasEndedBy(id; A2,E,A1,t,atrib) THEN 'activity' ∈ typeOf(A2) AND 'entity' ∈ typeOf(E) AND 'activity' ∈ typeOf(A1)
wasDerivedFrom	IF wasDerivedFrom(id; E2, E1, A/-, idwasGeneratedBy/-, idUsed/-, atrib) THEN 'entity' ∈ typeOf(E2) AND 'entity' ∈ typeOf(E1) AND 'activity' ∈ typeOf(A)
wasRevisionOf	IF wasDerivedFrom(...,[prov:type='prov:Revision']), THEN alternateOf(E2,E1).
wasAttributedTo	IF wasAttributedTo(id; E,Ag,atrib) THEN 'entity' ∈ typeOf(e) AND 'agent' ∈ typeOf(Ag)
wasInformedBy	IF wasInformedBy(id; A2,A1,atrib) THEN ∃ E, idwasGeneratedBy, t1, idUsed, & t2, ⟹ wasGeneratedBy(id; E,A1,t1,atrib) and used(id; A2,E,t2,atrib)
actedOnBehalfOf	IF actedOnBehalfOf(id; Ag2,Ag1,A,atrib) THEN 'agent' ∈ typeOf(Ag2) AND 'agent' ∈ typeOf(Ag1) AND 'activity' ∈ typeOf(A)
Nova wasAssociatedWith	IF wasAssociatedWith(id; E, A, Ag) THEN 'entity' ∈ typeOf(E) AND 'activity' ∈ typeOf(A) AND 'agent' ∈ typeOf(Ag) ⟹ ∃ wasStartedBy || ∃ wasEndedBy || ∃ used AND ∃ wasAttributedTo

2.3.2.1 Extensões do PROV

Apesar do modelo PROV ser mais amplo que o OPM, ainda é um modelo para proveniência de dados genérico. Com isso, surgiram extensões do modelo para refletir melhor a proveniência, sobre alguma perspectiva específica.

São exemplos dassas extensões o ProvONE (72), que é um algoritmo capaz de se gerar grafos, que demostram a proveniência prospectiva e retrospectiva de experimentos científicos, facilitando a representação do conhecimento, e o MetaStore (73), que é um *framework* de gerenciamento de metadados para repositórios de dados científicos (*e-Science*), onde os dados são armazenados em bases *NoSQL*, permitindo lidar com padrões de metadados heterogêneos.

Esses trabalhos refletem parte da abordagem que será apresentada no capítulo 4.

2.4 Comparação *Logs* x OPM X PROV

O modelo PROV é mais extenso que o OPM, que por sua vez, representa melhor as derivações e acontecimento do que um sistema de *Logs*. 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 2.2 resume a comparação entre os modelos PROV e OPM.

Tabela 2.2: 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

Para o uso em sistemas multiagentes o modelo PROV leva vantagens, visto que o modelo tem por objetivo armazenar dados de procedência de forma detalhada, focalizando as responsabilidades dos agentes em cada item de proveniência como explicado por (46).

3
Revisão *quasi* Sistemática

Segundo (74) , uma revisão *quasi* sistemática é um estudo secundário que tem como objetivo identificar, avaliar e interpretar os trabalhos existentes na literatura. Sua diferença com a revisão sistemática tradicional, é a caracterização principal do estudo, onde na *quasi* revisão é feita a caracterização de uma área, ou a não existência de objetos para comparação, conforme (76).

Na próxima seção abordaremos a metodologia utilizada durante a Revisão *quasi* Sistemática e na sequência os trabalhos relacionados.

3.0.1
Metodologia

Para embasar o trabalho desenvolvido nesta tese, foi realizada uma revisão *quasi* sistemática, baseada em um protocolo definido. Este protocolo apresenta o tema a ser investigado de uma forma específica, assim como as instruções para a seleção, análise e resumo de trabalhos relevantes, conforme (75).

A seleção dos artigos, considerou como critérios de inclusão os artigos que possuíssem proveniência de dados e sistemas multiagentes, e de exclusão, se não tivesse ligação entre ambos, dessa forma, foi realizada a seleção em 5 etapas, que são:

1. Busca e Catalogação: A *String* de busca (Tabela 3.1) aplicada no estudo, foi formada pelas palavras-chave apresentadas na Tabela 3.2, e executada sobre as bases listadas na Tabela 3.3. Com isso, chegou-se ao total de artigos por base, conforme Figura 3.1. Os artigos recuperados foram catalogados utilizando o *Parsifal*[1], que auxiliou nas etapas seguintes.

Tabela 3.1: String de busca.

(“multiagent systems” OR “multi-agent systems” OR “autonomous agents” OR “autonomous systems” OR “software agents”) AND (“data provenance” OR “provenance”)

[1] *Parsifal* - Disponível em: <https://parsif.al/>

Tabela 3.2: Palavras-chave do estudo.

Palavras-chave	Sinônimo
data provenance	provenance
multi-agent systems	autonomous agents autonomous systems multiagent systems software agents

Tabela 3.3: Bases de busca.

Base	URL
ACM Digital Library	<http://portal.acm.org>
El Compendex	<http://www.engineeringvillage.com>
IEEE Digital Library	<http://ieeexplore.ieee.org>
ISI Web of Science	<http://www.isiknowledge.com>
Science@Direct	<http://www.sciencedirect.com>
Scopus	<http://www.scopus.com>

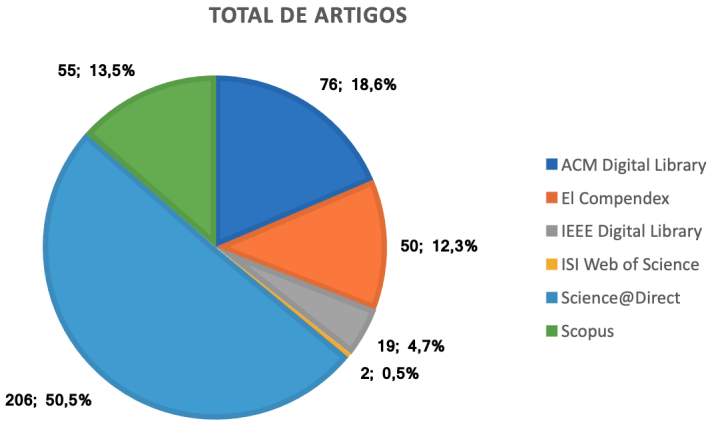


Figura 3.1: Total de artigos por base.

2. Eliminação de artigos repetidos: A ferramenta *Parsifal* possibilita a importação de arquivos *bibtex*, fornecidos diretamente pelas bases de buscas. Com isso, para evitar artigos repetidos, realizou-se a verificação dos arquivos importados de todas as bases, de forma unificada. A própria ferramenta já apresenta uma pré-avaliação de artigos duplicados, mas mesmo assim, foi feita uma busca manual. Como resultado desse filtro, tivemos um total de 68 artigos duplicados entre as bases, o que representa 14,3% conforme o gráfico da Figura 3.2.

   Já o total de artigos duplicados por base, pode ser vistos na Figura 3.3.

3. Seleção de artigos com base nos títulos: Com os artigos importados na ferramenta *Parsifal* e tendo eliminado os duplicados, para cada trabalho

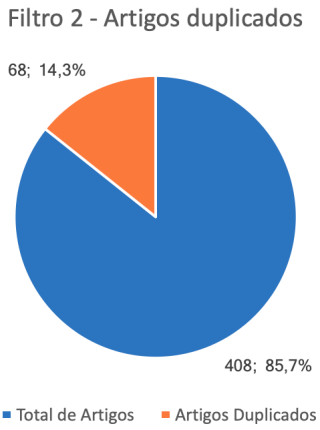
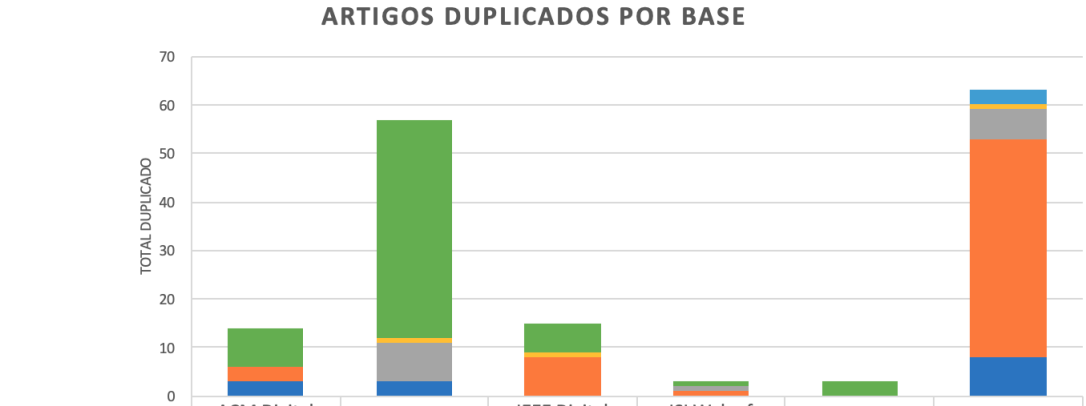


Figura 3.2: Artigos Totais vs Duplicados



	ACM Digital Library	EI Compendex	IEEE Digital Library	ISI Web of Science	Science@Direct	Scopus
Scopus	8	45	6	1	3	0
Science@Direct	0	0	0	0	0	3
ISI Web of Science	0	1	1	0	0	1
IEEE Digital Library	0	8	0	1	0	6
EI Compendex	3	0	8	1	0	45
ACM Digital Library	3	3	0	0	0	8

Figura 3.3: Artigos duplicados por base.

foi verificado em seu título ligação com o trabalho desta tese, onde os que não se enquadravam completamente no contexto da pesquisa foram eliminados. Para evitar falso positivo, alguns artigos foram submetidos a leitura dos resumos, que é o próximo filtro.

4. Seleção de artigos com base na leitura dos resumos: Os resumos dos artigos selecionados através da análise do título foram verificados seguindo os critérios de exclusão a seguir:

   i O trabalho possuía ligação com sistemas autônomos.

   ii O trabalho utilizava proveniência de dados.

iii No trabalho havia ligação da proveniência de dados com o sistema autônomo.

Caso o artigo não atendesse a um desses critérios era eliminado da seleção. Os resultados dos artigos selecionados com base neste filtro, podem ser vistos nas figuras 3.4 e 3.5 a seguir, que representam o número de artigos selecionados para leitura completa versos o total de artigos recuperados pela busca, e o total de artigos selecionados por base, respectivamente.

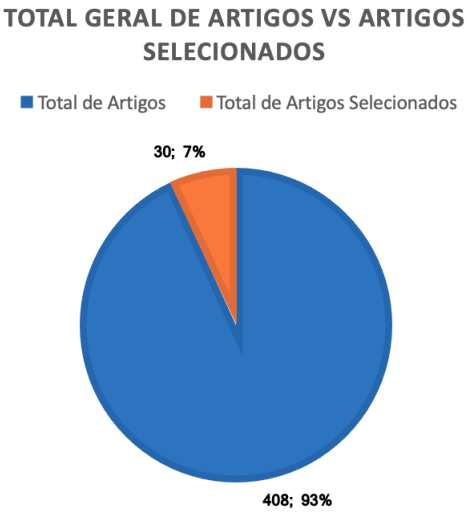


Figura 3.4: Artigos selecionados no geral.

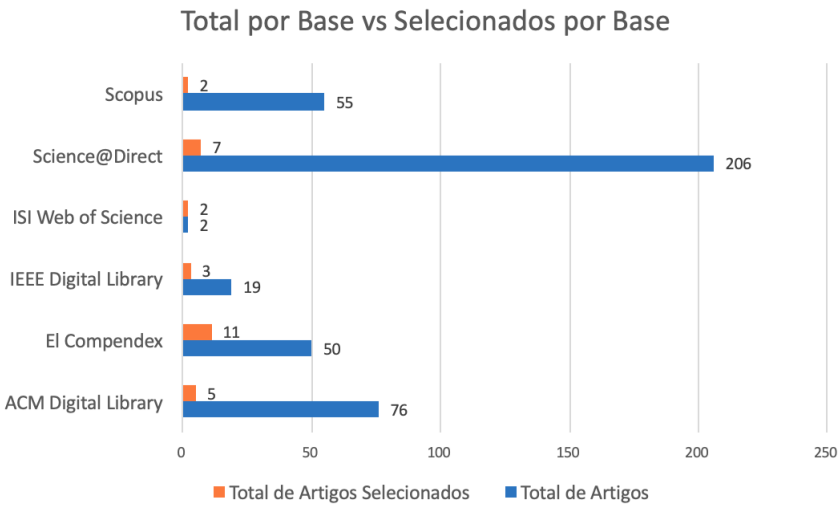


Figura 3.5: Artigos selecionados por base.

5. Seleção das publicações relevantes com base na leitura completa do artigo: Neste filtro, as publicações selecionadas foram lidas por completo e analisadas usando os critérios de exclusão abaixo, visto que se trata

de uma análise mais profunda e as soluções podem contribuir com este trabalho.

- O trabalho utiliza algum modelo de proveniência de dados?
- O artigo trata da proveniência dos dados em sistemas autônomos?
- O modelo de proveniência interage com o sistema autônomo?

Os resultados deste filtro serão apresentados na próxima seção, como sendo os trabalhos relacionados. Como resultado desta revisão *quasi* sistemática, observou-se que o tema ainda é pouco explorado dentro das pesquisa em computação, mas que seus benefícios podem ser relevantes para as pesquisas que envolvem sistemas autônomos.

Poder rastrear e ter a possibilidade de compreender como determinadas ações ocorrem dentro de um sistema autônomo, garante mais confiabilidade aos sistemas desenvolvidos e agrega mais segurança a sua tomada de decisão.

3.1 Trabalhos Relacionados

Esta seção tem como objetivo apresentar os principais trabalhos que servem de base e inspiram esta proposta, seguindo os resultados obtidos pela revisão *quasi* sistemática da seção anterior. Tais trabalhos envolvem proveniência de dados e sistemas autônomos.

No trabalho de (56), é apresentado uma proposta onde dados de saúde são registrados por meio de um sistema autônomo, e este é responsável por representá-lo em forma de proveniência, para deixar claro o histórico de tratamento dos pacientes e auxiliar a equipe médica na tomada de decisão. Apesar do trabalho envolver o uso de proveniência e sistemas autônomos, o foco está nos dados do paciente, diferente desta proposta que está sobre o próprio sistema autônomo. Além disso, o sistema desenvolvido, constrói seu próprio modelo de proveniência.

O trabalho de (22) aborda a construção de sistema autônomos explicáveis, ou seja, que documentam suas ações e permitem compreender o raciocínio dos agentes. Assim como o trabalho anterior, o mesmo não aborda o uso de modelos de proveniência como forma de armazenar e permitir a reprodutibilidade, rastreamento ou tornar explicáveis as ações. Apesar de nenhum formalismo na coleta e armazenamento dos dados para análise, utiliza como proposta uma base centralizada para os dados coletados, o que em sistemas distribuídos como o caso dos SMA, torna a análise mais fácil do ponto de vista do usuário. Esse trabalho possui forte ligação com esta proposta, com base no objetivo de tornar sistema autônomos explicáveis e por utilizar uma base de dados para

armazenar os dados de proveniência. Entretanto, o trabalho não formaliza um modelo de proveniência, utilizando o OPM ou PROV, além disso, não trata o problema de uma base centralizada de dados e sua comunicação, que serão discutidos no próximo capítulo.

Já o trabalho de (16), sobre modelagem de dados em sistemas autônomos e o de (13), denominado *AgentPrIMe*, pode ser considerado um trabalho chave na proposta, onde os autores abordam a modelagem de dados em sistemas autônomos e, justificam sua necessidade como uma forma de documentar os acontecimentos nas aplicações. O trabalho apresentou as questões levantadas na seção 1.2 desta proposta e um modelo para descrever as ações de sistemas autônomos. Entretanto, esse trabalho é de 2007 e, por isso, não baseia-se em modelos de proveniência de dados atuais e amplamente difundidos como o OPM ou o Prov, limitando a solução de modelagem ao caso apresenta no trabalho.

Outro trabalho interessante a esta proposta é o de (54), onde aborda-se o modelo 5W1H, apesar de levantar pontos interessantes sobre estruturas para a especificação, verificação, implementação e prototipagem de agentes e SMA, suas características diferem a desta proposta, por não focar no uso de proveniência de dados como método de rastreamento e explicação de ações em sistema autônomos, mas serve de inspiração e como forma de apoio as questões levantadas por este trabalho, assim como o trabalho de (86).

No trabalho de (77), é proposto um sistema autônomo de rastreamento de proveniência, separando o rastreador da aplicação, sendo que o rastreamento é feito por meio de agentes autônomos. Dessa forma, existem agentes no sistema que são responsáveis por rastrear as comunicações dos agentes da aplicação e registrar os dados. Nesse trabalho é necessário que os agentes se comuniquem através do modelo de quadro negro, conforme explicado em 2.1, o que não deixa claro o raciocínio dos agentes da aplicação, mas apenas os que foi coletado entre as conversas. Além disso, o trabalho não deixa claro como os dados são preservados seguindo algum modelo de proveniência.

Os trabalhos de (78) e (79), descrevem sistemas autônomos de simulação e abordam como o uso de proveniência pode auxiliar no refinamento do volume de dados. Apesar de aplicarem o modelo de proveniência PROV associada a sistemas multiagentes, seu uso é para coletar dados de simulação e não sobre os agentes empregados. O ponto chave dele é o uso do modelo PROV para armazenamento dos dados de proveniência.

Por fim, nos trabalhos (80) e (81), são apesentados o *ProvMASS*, construído sobre a biblioteca MASS (*Multi-Agent Spatial Simulation*), é uma estrutura que combina sistemas multiagentes voltados para simulação, com pro-

veniência de dados. Nessa abordagem as simulações são feitas com o uso de agentes de software, e por meio da proveniência é analisado o comportamento dos agentes. O foco do trabalho está em sistema de simulação de evacuação de ambientes. O comportamento dos agentes são analisados para verificar sua decisão em caso de emergência, e não abordam as decisões dos agentes ou visam identificar o porquê tomaram determinadas decisões.

Esses trabalhos apresentam um esboço da base dessa proposta, onde o trabalho desenvolvido baseia-se em abordagens já consistentes, melhorando suas lacunas e abordando uma nova visão a um problema importante.

4
Proveniência de Dados em Sistemas Multiagentes

A proveniência dos dados nos permite acompanhar as ações do sistema autônomo e sua interação com outros sistemas e humanos. Para isso, 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. Isso envolve os dados de entrada dos eventos, a sequência de interações que ocorreram e os resultados gerados. Uma visão geral do modelo PROV pode ser visto na Figura 4.1 (conforme apresentado por (91)), e envolve todas essas características supracitadas.

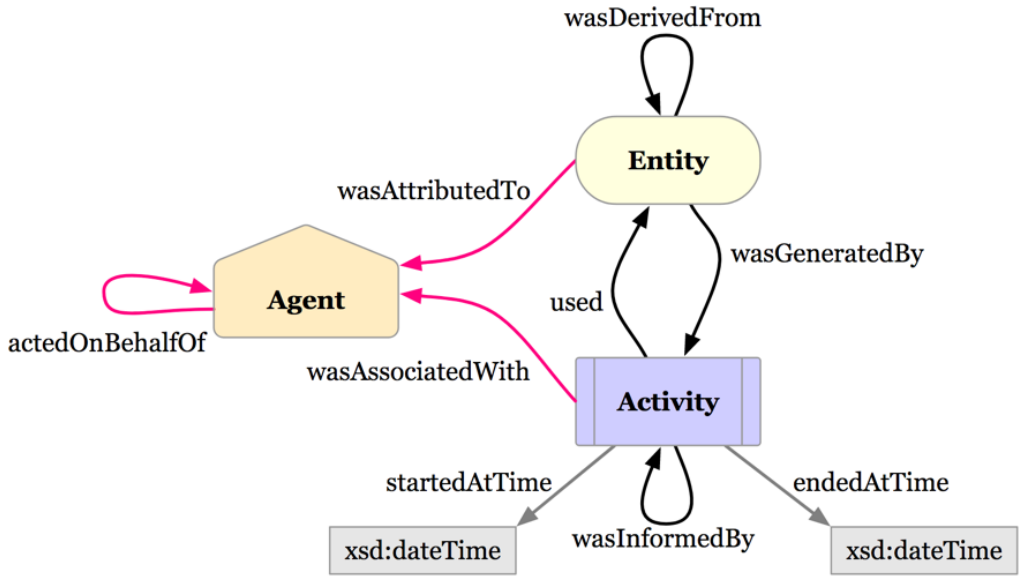


Figura 4.1: Modelo PROV W3C

Nesse trabalho, defendemos o uso do modelo de proveniência PROV, pois com o uso de *logs* existe a perda de conhecimento por não representar a ligação entre os eventos, pois os *logs* em muitos casos são limitados a linhas inseridas no meio do código que registram um determinado acontecimento ou passagem por um ponto. Um exemplo disso pode ser visto na Listagem 4.1, do código extraído da classe *BDIAgent.java* do BDI4JADE (41). Nesse exemplo temos o uso do *trace* para registrar qual trecho de código já foi executado dentro do *Behaviour* do agente, entretanto, como pode ser observado no trecho de código, não são registradas informações sobre a qual agente esse comportamento pertence,

quem executou essa chamada, o tempo que durou e os processos intermediários que foram executados para cumprir com essa ação, o que torna a utilidade dos *logs* baixa, uma vez que não explica claramente o comportamento do agente no software.

```
@Override
public void action() {
    log.trace("Beginning BDI-interpreter cycle.");
    log.trace("Reviewing beliefs.");
    beliefRevisionStrategy.reviewBeliefs(BDIAgent.this);
    synchronized (intentions) {
        Map<Goal, GoalStatus> goalStatus = new HashMap<
           Goal, GoalStatus>();
        Iterator<Intention> it = intentions.iterator();
        while (it.hasNext()) {
            Intention intention = it.next();
            GoalStatus status = intention.getStatus();
            switch (status) {
            case ACHIEVED:
            case NO_LONGER_DESIRED:
            case UNACHIEVABLE:
                intention.fireGoalFinishedEvent();
                it.remove();
                break;
            default:
                goalStatus.put(intention.getGoal(),
                   status);
                break;
            }
        }
        Set<Goal> generatedGoals =
           optionGenerationFunction
                .generateGoals(goalStatus);
        Set<Goal> newGoals = new HashSet<Goal>(
           generatedGoals);
        newGoals.removeAll(goalStatus.keySet());
        for (Goal goal : newGoals)
            {
                addGoal(goal);
            }
        Set<Goal> removedGoals = new HashSet<Goal>(
           goalStatus.keySet());
```

```
33          removedGoals.removeAll(generatedGoals);
34          for (Goal goal : removedGoals) {
35              it = intentions.iterator();
36              while (it.hasNext()) {
37                  Intention intention = it.next();
38                  if (intention.getGoal().equals(goal)) {
39                      intention.noLongerDesire();
40                      intention.fireGoalFinishedEvent();
41                      it.remove();
42                  }
43              }
44          }
45          goalStatus = new HashMap<Goal, GoalStatus>();
46          for (Intention intention : intentions) {
47              goalStatus.put(intention.getGoal(), intention
                    .getStatus());
48          }
49          Set<Goal> selectedGoals = deliberationFunction.
                filter(goalStatus);
50          log.trace("Selected goals to be intentions: "+
                selectedGoals.size());
51          for (Intention intention : intentions) {
52              if (selectedGoals.contains(intention.getGoal
                    ())) {
53                  intention.tryToAchive();
54              } else {
55                  intention.doWait();
56              }
57          }
58          if (intentions.isEmpty()) {
59              log.trace("No goals or intentions - blocking
                    cycle.");
60              fireStateIdle();
61              this.block();
62          }
63      }
64      log.trace("BDI-interpreter cycle finished.");
65  }
```

Listing 4.1: Uso de *logs* no BDI4JADE

Uma possibilidade para termos um melhor registro dos eventos é com o uso de um modelo de proveniência como o PROV, visto que o mesmo já é projetado para registrar informações desse tipo e possibilita a análise por diversos métodos.

Entretanto, uma questão que persiste é; como capturar esse dados de modo que a arquitetura do software não seja afetada e o sistema não fique poluído com o uso excessivo de código inserido para capturar todas essas informações?

Na seção a seguir (4.1), abordaremos uma solução viável, utilizada neste trabalho.

4.1 Proposta

Atualmente um paradigma muito utilizado na construção de software é o de orientação a objeto (OO), que possibilita que na programação seja utilizada herança, polimorfismo e interfaces. Entretanto, uma característica importante a ser observada é com relação a sintaxe e a semântica.

A sintaxe é o formato utilizado pelas linguagens de programação para definir o funcionamento do código, ou seja, a estrutura que o código deve possuir. Já a semântica indica como o software deve se comportar quando executadas num determinado ambiente. Esse conhecimento é importante pois quando os códigos do software são compilados, temos a garantia que apenas a sintaxe está correta. Isso nos leva a muitas vezes para inspecionar o código por meio do uso de depuradores, pontos de interrupção e uso de *logs*.

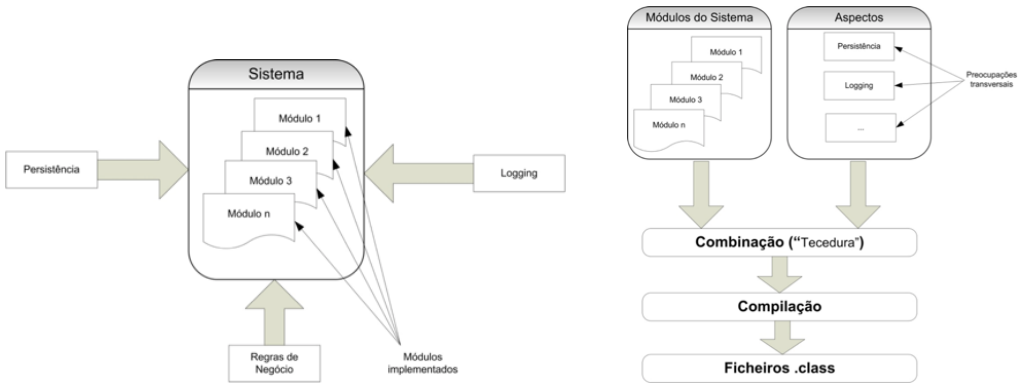
Como já apresentado, o uso de *logs* pode não ser uma boa opção para captura de informações em sistemas complexos, como é o caso de software que usam alguma inteligência artificial distribuída, tal como um SMA. Já o uso de uma abordagem usando aspecto pode agregar recursos em uma programação OO, como abordado por (49).

Uma abordagem comum para o desenvolvimento de softwares complexos é a modularização, onde conforme (50), torna o software mais flexível e pelos módulos serem de certo modo independentes, facilita na compreensão do sistema. Contudo, registrar as ações nesses sistemas pode ser um desafio, visto que podem ser distribuídos, apresentar espalhamento de código ou emaranhamento entre os módulos. Outro ponto que torna a gestão dos *logs* um desafio maior é a integração entre eles, que também deveriam compor os registros, dado que o software é o mesmo, como representado na Figura 4.2(a), conforme (51).

Para o isolamento e a modularização de questões cruzadas, em 1997 foi apresentada a Programação Orientada a Aspectos (POA) (82). Do ponto de vista da Engenharia de Software, a POA permite o suporte e o aprimoramento dos princípios de *Design* Orientado a Objetos (DOO), como o Princípio da Única Responsabilidade (PUR) e o Princípio Aberto-Fechado (PAF), apresentados por (83).

Devido à separação de responsabilidades, o POA permite um alto nível de coesão e baixo acoplamento entre módulos e aspectos e são esses os princípios fundamentais e necessários para alcançar um projeto de software modular, conforme (84).

Nesses casos, a proveniência pode ser implementada usando aspectos, onde não será necessário modificar a arquitetura do sistema multiagente para a inserção da captura dos traços de procedência. Dessa forma a proveniência é tratada pelo aspecto como uma preocupação transversal, interceptando os pontos de interesse identificáveis pela POA. Conforme (92), na Figura 4.2(a) é feita a interceptação dos pontos e adicionado o aspecto ao código do software no processo de compilação, conforme a Figura 4.2(b).



4.2(a): Interceptação 4.2(b): Integração

Figura 4.2: Interceptação dos dados e Integração da captura ao SMA.

Com isso, o código da plataforma de agentes e do software não sofre nenhuma alteração na sua estrutura, e a proveniência atua nos pontos de execução identificáveis entre as funções, registrando cada acontecimento. Essa captura deve ser feita para os agentes, para o sistema multiagente, conforme a Figura 4.3 e também dentro do BDI, conforme a Figura 4.4.

Para realizar esse trabalho de coleta e armazenamento dos dados de proveniência, seguindo o modelo PROV, voltado para o uso em sistemas autônomos, foi desenvolvido o *Framework* FProvW3C(46), que será apresentado na próxima seção (4.2).

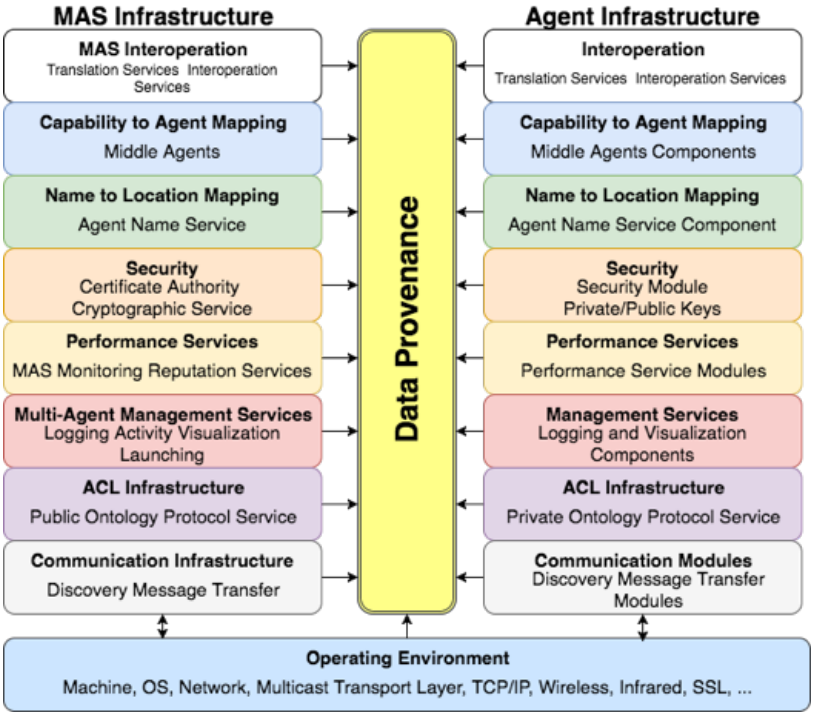


Figura 4.3: Captura dos dados no SMA

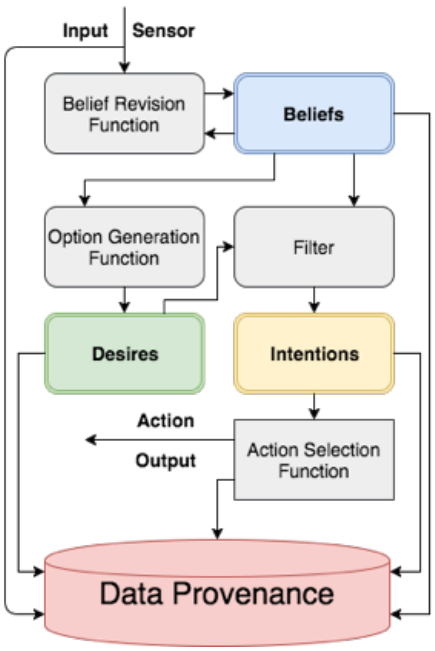


Figura 4.4: Captura dos dados no BDI

4.2 *Framework* FProvW3C

Dispensar do programar a função de inserir a captura da proveniência em meio ao seu código, reduz seu trabalho, evita a captura dos dados de maneira equivocada, restringe que os detalhes relevantes estejam de fato sendo capturados e corretamente armazenados.

Seguindo isto, foi desenvolvido um *Framework* para capturar a proveniência dos dados, denominado FProvW3C (46). Proposto e instanciado em um sistema autônomo que utiliza agentes para classificação de gases expelidos por humanos, conforme apresentado em (26). A experiência obtida com o desenvolvimento deste sistema levou à especificação do método 5W2H, para determinar os dados que devem ser registrados para modelar a proveniência dos dados.

Esse método (5W2H) é baseado nos trabalhos de (54) e (47), onde busca suporte à identificação dos dados, processos e ações mais importantes dentro de um sistema autônomo. 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 o responsável pela ação fez e por que o fez, auxiliando no mapeamento do sistema multiagentes para o modelo PROV.

As sete questões do método 5W2H para sistemas autônomos são:

- O que o agente fez?
- Quem (ou qual agente) fez isso?
- Onde aconteceu (ação ou falha)?
- Quando a ação ou falha ocorreu?
- Por que a ação ou falha ocorreu?
- Como a ação ou falha ocorreu?
- Quanto isso custou ao sistema?

Embora simples, o método 5W2H apoia a análise e a aquisição de conhecimento sobre um determinado processo, problema ou ação que ainda não foi realizado ou que já ocorreu. As perguntas servem para extrair as noções de proveniência central, de acordo com (87) apud (86), ou seja, evento, espaço, ação no tempo e agente, essa relação pode ser vista na Tabela 4.1 adaptada de (86).

Tabela 4.1: Mapeamento entre o 5W2H as questões de proveniência.

Notação	Questão
Evento	O que será feito (etapas - What)
Espaço	Onde será feito (local - Where)
Tempo	Quando será feito (tempo) - When)
Acão	Como será feito (método - How), Por que será feito (justificativa - Why), Quanto custará fazer (custo - How much)
Agente	Por quem será feito (responsabilidade - Who)

Quando o 5W2H é aplicada à SMA, pode auxiliar de três formas diferentes, conforme apresentado a seguir:

- Diagnóstico: O método 5W2H permite a investigação de um problema ou processo, a detecção de falhas e pontos de vulnerabilidade. Para lidar com esses problemas, é necessário aumentar a quantidade de informações do sistema e, nesse caso, o *Framework* do FProvW3C pode ser usada.
- Plano de ação: O processo de validação de um sistema autônomo não é uma tarefa trivial (44), porque o agente deve ter um comportamento autônomo mesmo seguindo normas. A proveniência dos dados auxilia na construção de uma base de informações sobre os agentes, onde é possível por exemplo: i) verificar seu plano de ação (suas decisões) sobre o que deve ser feito e ii) eliminar comportamentos irregulares que possam causar alguma anomalia no sistema e o *Framework* FProvW3C pode auxiliar nesse processo.
- Documentação: O uso de proveniência possibilita documentar as etapas do sistema, extrair relatórios de execuções, falhas, pontos de vulnerabilidade e resultados produzidos por uma determinada entrada ou saída, onde o *Framework* FProvW3C também pode ser utilizado.

O método 5W2H permite dividir os dados em diferentes partes, que por sua vez mostravam quais pessoas, ou agentes de software, estavam operando em cada situação e cada ação. Mostra também em que momento certas atividades estavam acontecendo, qual a sequência de ações que produziu o resultado, como a ação foi executada e quais recursos foram utilizados. Esse mapeamento, pode ser observado na Figura 4.5, adaptada de (86).

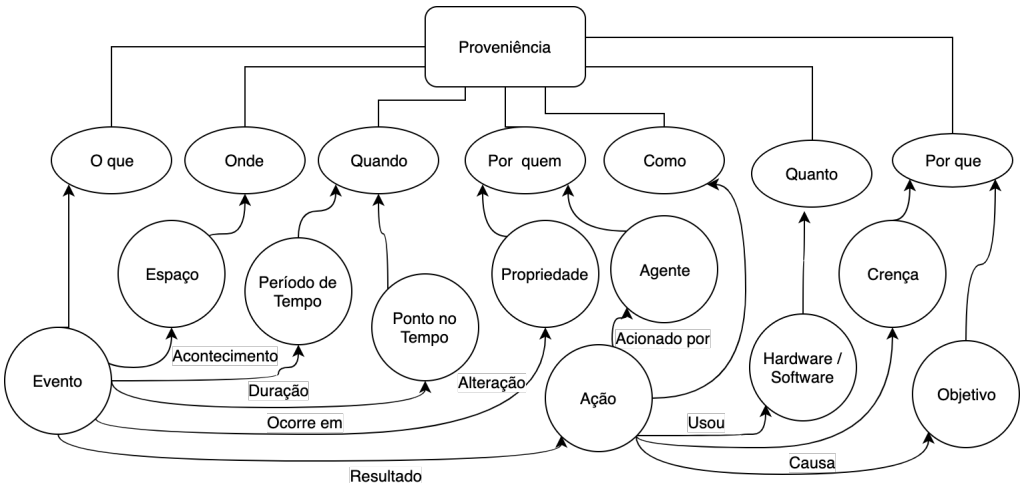


Figura 4.5: Captura modelada com o 5W2H.

Alinhando as questões apresentadas pelo método 5W2H com o uso de proveniência de dados em sistemas autônomos, como resultado, é possível capturar de forma dinâmica o comportamento do sistema e responder às questões levantadas na Seção 1.2.

Um mapeamento entro o modelo PROV e o 5W2H, pode ser visto na Tabela 4.2.

Tabela 4.2: PROV mapeado com o 5W2H

PROV	5W2H
Entity	Who
Activity	What
Agent	Who
used	How, Why
wasGeneratedBy	What, Where, Who
wasAssociatedWith	What, Where, Who
wasStartedBy	When, Who, What
wasDerivedFrom	Where, Why, When
wasEndedBy	When, Who, What
wasRevisionOf	Where, Why, When
wasAttributedTo	What, Where, Who
wasInformedBy	Where, Why, When
actedOnBehalfOf	How much, Why, Who

Entretanto, ressalta-se que a primeira versão do *Framework* FProvW3C[1] foi baseada em um modelo relacional de dados, apresentado em (26), baseava-se em código inserido diretamente dentro do SMA e armazenava os dados em um SGDB (Sistema Gerenciador de Banco de Dados) Relacional, a saber o MySQL. Nessa abordagem foram identificadas uma série de limitações que serão apresentadas na seção à seguir.

4.2.1 *Framework* FProvW3C Relacional

O *Framework* FProvW3C trabalha com uma abordagem ansiosa (48), para que os dados sejam coletados a todo momento e possam ser consultados a seguir. Atualmente, em sua arquitetura (Figura 4.6), o *Framework* FProvW3C apresenta toda a especificação do PROV-DM com as anotações em *Java Persistence API* (JPA).

Além da modelagem, o *Framework* FProvW3C traz anotações de persistência. Isso reduz o trabalho dos usuários e evita erros de mapeamento por aqueles que não têm extenso conhecimento do PROV W3C. Por ser um *Fra-*

[1]FProvW3C - Disponível em: https://github.com/tassioferenzini/FrameworkFProvW3C

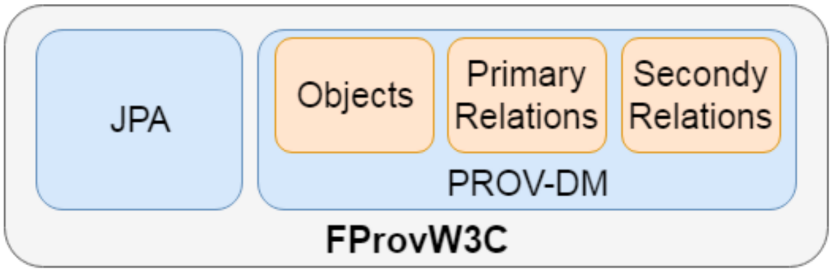


Figura 4.6: Arquitetura do FProvW3C

mework, torna possível integrá-lo em diferentes sistemas. Os *frozen-spots* do *Framework* são os vértices principais e os *hot-spots* são as relações causais, ambas definidas pelo modelo PROV, e se adaptam a diferentes contextos de aplicação.

O FProvW3C é uma estrutura de mapeamento de objetos relacionais, encarregada de criar o banco de dados, as tabelas e seus respectivos atributos no SGBD. O modelo PROV determina as classes e como elas se relacionam, além dos atributos básicos. Desta forma, o *Framework* fornece as classes com os atributos básicos (*frozen-spots*) e, além disso, permite a criação de *hot-spots*, estendendo os atributos de cada classe e os relacionamentos. Estes atributos destinam-se a representarem as características do sistema no qual o FProvW3C será aplicado.

Como o *Framework* executa mapeamento de dados de acordo o PROV, seu uso é facilitado para o usuário, onde é possível aplicar a proveniência dos dados em suas aplicações. O FProvW3C cria uma camada intermediária entre o aplicação e banco de dados. Essa camada permite tratar e mapear os dados a serem armazenados, vinculando a fonte de informações a eles, conforme apresenta na Figura 4.7, extraída de (26). Portanto, todos os dados manipulados ou ações realizadas pelo sistema multiagente podem ser catalogadas.

Um exemplo de uso do FProvW3C em uso no *Smell App*, pode ser visto no trecho de código extraído da classe “AlertAgent.java”, conforme Listagem 4.2.

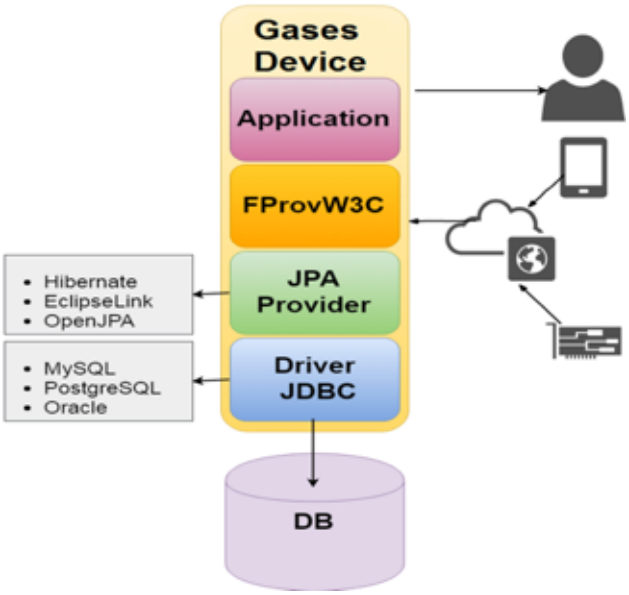


Figura 4.7: FProvW3C aplicado no sistema de análise de Gases.

```
protected void setup() {
  //FProvW3c
  AgentP agDevice = new AgentP();
  // agDevice.setIdUser(user.getId());
  agDevice.setTypeAgent("Software Agent - Alert");
  new AgentDAO().save(agDevice);
    addBehaviour(new TickerBehaviour(this, 100000) {
    @Override
    protected void onTick() {
      System.out.println("Estou vivo");
      //FProvW3c
      ActivityN ac = new ActivityN();
      Timestamp startT = new Timestamp(System.
         currentTimeMillis());
      ac.setStartTime(startT);
      List<Report_Gases> list_report_gases = new
         ReportGasDAO().getAll();
        for (Report_Gases list_report_gase :
           list_report_gases) {
           double methane = list_report_gase.
              getPercentmethane();
           double hydrogen = list_report_gase.
              getPercenthydrogen();
           if(methane>hydrogen){
              int reportID = list_report_gase.getReport()
```

```
                .getId();
21              int userId = list_report_gase.getReport().
                   getUser().getId();
22              ActivityN ac2 = new ActivityN();
23              startT = new Timestamp(System.
                   currentTimeMillis());
24              ac2.setStartTime(startT);
25              Timestamp endT = new Timestamp(System.
                   currentTimeMillis());
26              ac2.setEndTime(endT);
27              ac2.setDescription("Alert Agent sent an
                   alert to User "+ userId+ " to seek a
                   medical assistance: methane > hydrogen.
                   (Report Id" +reportID+")");
28              new ActivityDAO().save(ac2);
29            }
30        }
31      //FProvW3c
32      Timestamp endT = new Timestamp(System.
           currentTimeMillis());
33      ac.setEndTime(endT);
34      ac.setDescription("Alert Agent verified data from
           all patients to generate alerts.");
35      new ActivityDAO().save(ac);
36    }
37  });
38 }
```

Listing 4.2: Uso do FProvW3C no Smell App.

Como nessa versão tivemos os problemas vinculados à; i) alta acoplamento, ii) escalabilidade horizontal limitada, pois o banco de dados era central e iii) gargalo no processamento dos dados por causa da comunicação da rede com o banco e o processamento do banco. Por conta disso, foi desenvolvido uma nova versão do *Framework*, com uma nova abordagem (Seção 4.2.2) resolvendo os problemas desta versão Relacional.

4.2.2 *Framework* FProvW3C *NoSQL*

A programação orientada a aspectos (POA) por meio do AspectJ (85), permite que os chamados aspectos transversais sejam modularizados pelos códigos implementados em forma de aspectos. Geralmente, esses códigos

representam preocupações com métodos ortogonais, sendo no geral capazes de respeitar os princípios do *design* orientado a objetos (83).

O AspectJ é uma extensão simples e prática para Java, que fornece suporte para implementação modular de uma variedade de preocupações transversais e que é compilado no *bytecode* padrão do Java.

Ressalta-se que a modularização da interação de classe por meio de aspectos, geralmente resulta em soluções de software complexas para evoluir, pois os aspectos associados podem não refletir a estrutura e coesão dos recursos e módulos refinados pela interação de classe (83). Contudo, nesse trabalho os aspectos são aplicados sobre a plataforma de SMA, o que não afeta a aplicação autônoma, visto que o *Framework* está vinculado à plataforma e não à aplicação como na versão anterior.

A diferença da abordagem proposta aqui para a que foi utilizada em (26), apresentada anteriormente, é o acesso ao dados da plataforma do agente por meio de aspectos, que não é necessário a inserção de código para a captura da proveniência junto ao código original do software.

Com isso, a coleta e armazenamento de dados, seguindo a mesma arquitetura interna apresentada em 4.6 e mapeado pelo método 5W2H, pode ser feito com o *Framework* FProvW3C (46), que se baseia na programação orientada a aspectos para interceptar código de uma plataforma multiagente e registrar os dados de proveniência do sistema autônomo.

Um exemplo da captura dos dados de proveniência dentro de um sistema autônomo, por meio de aspecto, utilizando o FProvW3C NoSQL[2], pode ser visto na Listagem 4.3.

```
1  @Around("execution(* br.pucrio.inf.les.bdi4jade.core.
       BDIAgent.BDIInterpreter.action(..))")
2  public void PROVBDIAgentAction(ProceedingJoinPoint
       joinPoint) throws Throwable {
3      ProvActivity ac = new ProvActivity();
4      Timestamp startT = new Timestamp(System.
           currentTimeMillis());
5      ac.setStartTime(startT);
6      joinPoint.proceed();
7      Timestamp endT = new Timestamp(System.
           currentTimeMillis());
8      ac.setEndTime(endT);
9      ac.setDescription("BDI-interpreter cycle");
10     new ProvActivityDAO().save(ac);
```

[2]FProvW3C NoSQL - Disponível em: https://github.com/tassioferenzini/Framework FProvW3C-NoSQL

```
11      }
```

Listing 4.3: Capturando proveniência por meio de aspecto no BDI4JADE

Nesse exemplo da Listagem 4.3, está descrita a captura da atividade *action* apresentada na Listagem 4.1, onde são registrados os dados de início e fim da atividade e uma descrição à mesma. Além desses dados, para o exemplo apresentado na Listagem 4.1, deve-se capturar as chamadas que manipulam as crenças e os objetivos do agente e de seu processo deliberativo, referenciando a atividade que ocorreu, o agente que a executou e a entidade que foi impactada pela ação.

Nessa versão do *Framework*, temos uma arquitetura onde cada plataforma de agente possui sua própria base de dados e essas bases formam um *cluster* assíncrono. Dessa maneira, cada agente que roda em cada plataforma de modo distribuído, possui uma base de dados local, que por sua vez e interligada assincronamente com as bases de outras plataformas, formando uma base de proveniência única, auxiliando à explicar o comportamento do sistemas autônomos distribuídos.

Um esboço dessa arquitetura pode ser visto na Figura 4.8.

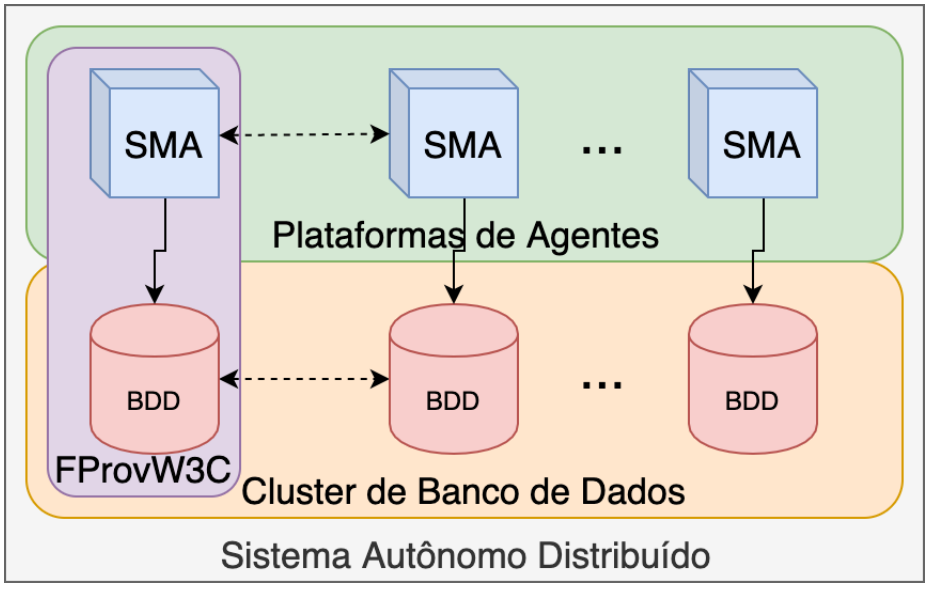


Figura 4.8: Arquitetura do FProvW3C NoSQL em um sistema autônomo.

Essa nova versão do *Framework* FProvW3C foi aplicada na plataforma BDI4JADE, que será apresentada na próxima seção 4.3.

4.3 Prov-DBI4JADE

Em sistemas autônomos, a proveniência depende dos componentes da aplicação que as registram, gerando a documentação adequada a partir da

qual a proveniência pode ser determinada. Conforme (16), demarcar e entender por que um determinado resultado surgiu é especialmente importante quando entidades autônomas estão envolvidas em um sistema, porque, nesses sistemas, a intenção por trás de qualquer processo que ocorre não é fixa no projeto original do sistema e, portanto, não pode ser entendido unicamente pela análise de um diagrama.

O JADE (Java Agent DEvelopment Framework) (36) é uma plataforma de desenvolvimento de sistemas baseados em agente, em conformidade com as especificações do FIPA (37) para sistemas multiagentes interoperáveis. Já o BDI4JADE (41) aproveita todos os recursos fornecidos pelo JADE e implementa a arquitetura BDI (crença-desejo-intenção) (10), com seu ciclo de raciocínio, tornando os agentes autônomos inteligentes.

A plataforma BDI4JADE (41) faz uso de *logs* para registrar seus eventos. No entanto, os sistemas de *logs* da linguagem JAVA não apresentam nenhuma informação detalhada, conforme já apresentado no capítulo 2.

Com o uso de aspecto pelo *Framework* FProvW3C, não é necessário modificar a arquitetura da plataforma multiagente do BDI4JADE para inserir os traços de proveniência. Como tal, a proveniência é tratada como uma preocupação transversal, de modo que a plataforma não sofra qualquer alteração na sua estrutura, e a proveniência atua nos pontos de execução identificáveis entre as funções, registrando cada evento. Como resultado, todas as ações do sistema autônomo e do seu ciclo de raciocínio são capturadas e registradas, conforme descrito em (46).

O *Framework* FProvW3C NoSQL trabalha com uma abordagem ansiosa, para que os dados sejam coletados a todo momento e possam ser consultados a seguir, assim como a versão relacional.

Atualmente, o *Framework* FProvW3C apresenta toda a especificação do PROV-DM com as anotações em JPA, conforme apresentado na seção 4.2.2. Como o *Framework* executa mapeamento de dados de acordo o PROV, seu uso é facilitado para o usuário, onde é possível aplicar a proveniência dos dados em suas aplicações sem ter a preocupação de como fazer isso. O FProvW3C cria uma camada intermediária entre o aplicação e banco de dados. Esta camada permite tratar e mapear os dados a serem armazenados, vinculando a fonte de informações a eles. Portanto, todos os dados manipulados ou ações realizadas pelo sistema autônomo podem ser catalogadas.

A fusão do FProvW3C com o BDI4JADE, criou o que denominamos PROV-BDI4JADE[3]. Onde os aspectos são construídos sobre o *Framework*

[3]PROV-BDI4JADE - Disponível em: https://github.com/tassioferenzini/Provenance-BDI4JADE

modelado para o PROV W3C e durante o processo de compilação agregados ao BDI4JADE, conforme Figura 4.9.

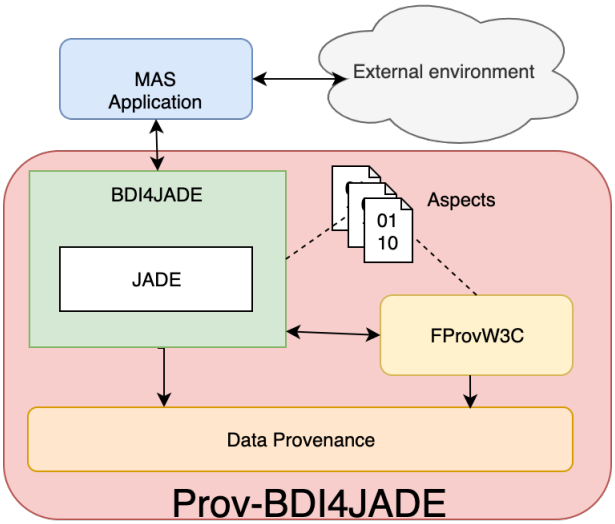


Figura 4.9: Arquitetura do Prov-BDI4JADE

Para identificarmos os pontos de interesse a serem coletados pela proveniência dentro da plataforma BDI4JADE, mapeou-se as classes estendidas, apresentados em (41). Nas figuras 4.10, 4.11, 4.12 e 4.13 temos destacados em cinza os principais pontos onde a proveniência atua dentro da plataforma, no core do agente BDI, nas crenças do agente, nos planos gerados e nas intenções de cada agente.

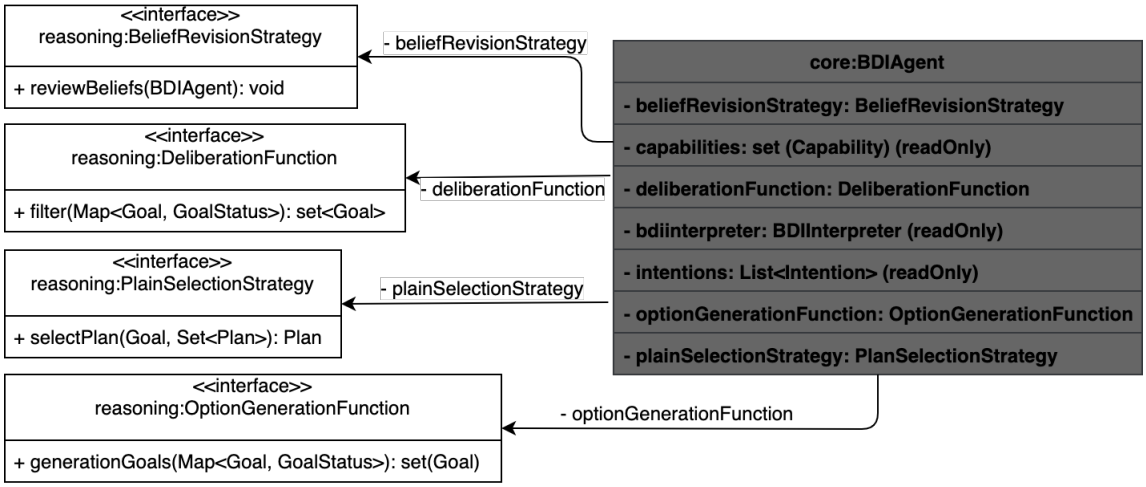


Figura 4.10: Proveniência no BDIAgent.

A aplicação do PROV-DBI4JADE na construção de sistemas autônomos, não exige que o desenvolvedor saiba em que ponto deve-se aplicar a captura de proveniência, visto que já estão definidos nos aspectos e no *Framework* FProvW3C. Essa ação facilita o trabalho do desenvolvedor e garante que sem

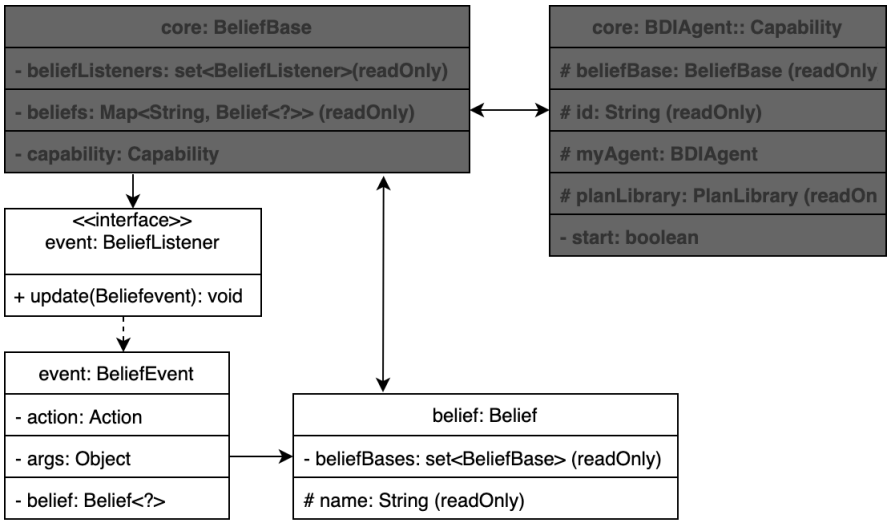


Figura 4.11: Proveniência nas Crenças.

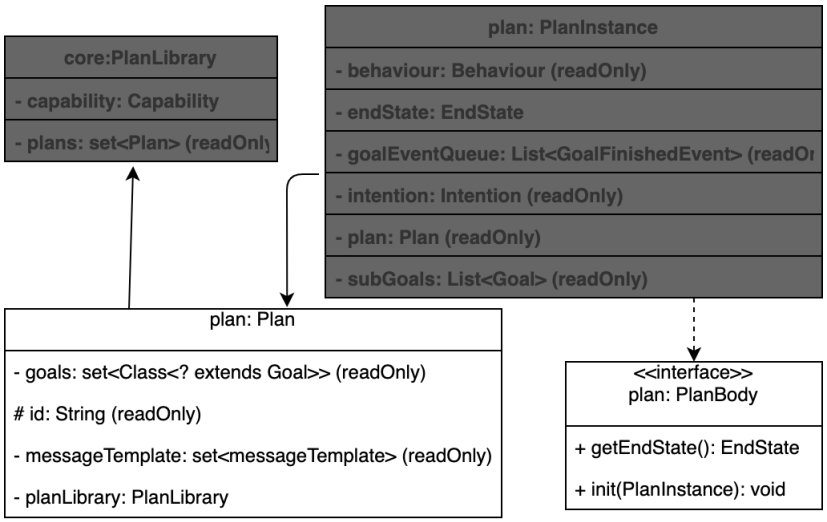


Figura 4.12: Proveniência nos Planos.

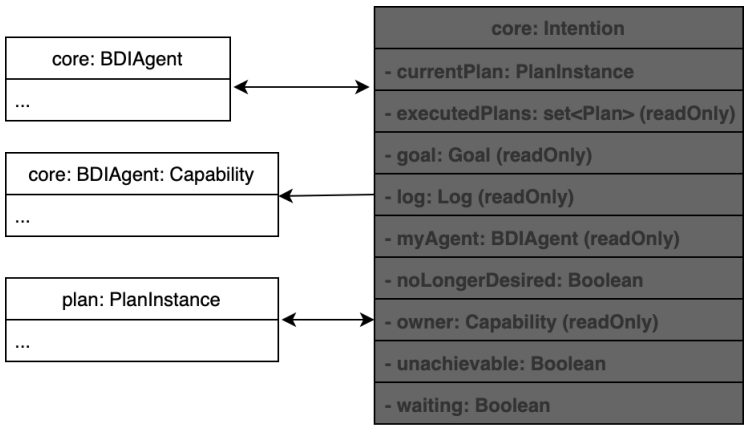


Figura 4.13: Proveniência nas Intenções.

a sua intervenção os dados são capturados nos pontos corretos, garantindo sua consistência e confiabilidade nas informações.

Além dos pontos destacados anteriormente, por meio do aspecto, ainda capturou-se as ações internas da plataforma BDI4JADE, os processos de instanciação e finalização dos sistemas analisados e as comunicações do entre os agentes autônomos, conforme apresenta o trecho de código a seguir (4.4), com a captura das mensagens trocadas dentro do sistema autônomo.

```
1  pointcut Setsend(ACLMessage arg1) :
2          execution(* jade.core.Agent.send(ACLMessage))  &&
3          args(arg1);
4
5      after(ACLMessage arg1) : Setsend(arg1) {
6
7          Date s = new Date();
8          activity = new ProvActivity();
9          activity.setNameActivity("Send Message " + arg1.
               getContent());
10         activity.setStartTime(s);
11         Date e = new Date();
12         activity.setEndTime(e);
13
14         agent = new ProvAgent();
15         agent.setTypeAgent(arg1.getSender().getClass().
               getSimpleName());
16         agent.setNameAgent(arg1.getSender().getName());
17
18         agent1 = new ProvAgent();
19         agent1.setTypeAgent(arg1.getAllReceiver().next().
               getClass().getSimpleName());
20         agent1.setNameAgent(arg1.getAllReceiver().next().
               toString());
21
22         act = new ProvActedOnBehalfOf();
23         act.setActivity(activity);
24         act.setAgent(agent);
25         act.setAgent1(agent1);
26
27         new Thread(t1).start();
28     }
```

Listing 4.4: Captura da comunicação do sistema.

Além disso, como o PROV-DBI4JADE é baseado na integração do FProvW3C e do BDI4JADE, caso o desenvolvedor tenha conhecimentos sobre

a plataforma de agentes e do modelo PROV da W3C, pode-se realizar algumas adaptações para melhorar ainda mais a granularidade dos dados capturados, quando a estrutura interna de um agente é definida, por meio dos *hots-spots* do *Framework*. Assim, os desenvolvedores podem considerar cada agente, e determinar como envolve-lo para registrar a proveniência sobre suas atividades.

Uma demostração de execução do Prov-BDI4JADE pode ser visto em: <https://youtu.be/S0XBZEYBCkQ>

Com isso, a proveniência de dados possibilita registrar informações em diferentes níveis de abstração, como um filme, e seu uso atrelado a programação por meio de aspectos, possibilita que os dados sejam capturados, sem que haja a necessidade de intervenção direta na arquitetura da plataforma de sistemas multiagentes.

O uso de aspectos se posiciona como uma abordagem viável para este caso, contudo, deve-se ter a consciência das consequências dessa abordagem, como identificar de maneira única os pontos de interseção. Essas questões serão tratadas na seção 6.2.

Na próxima seção (4.4) descreveremos alguns cenários de aplicação em que o Prov-DBI4JADE foi aplicado e outros cenários possíveis.

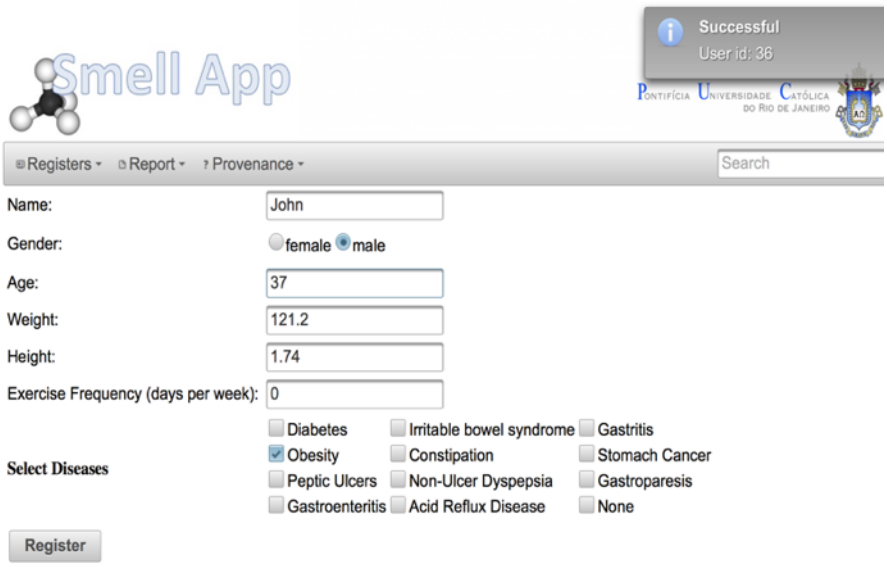
4.4 Cenários de Aplicação

Para descrever o primeiro cenário, podemos apresentar o caso demostrado em (26), onde o *Framework* FProvW3C foi empregado para criar uma camada intermediária entre o aplicativo e o banco de dados. Esta camada permite tratar e mapear os dados a serem armazenados, vinculando a fonte de informações a eles, assim como os passos executados pelos agentes. Com isso, o FProvW3C cria uma base de conhecimento baseada no uso do aplicativo, tornando-o explicativo.

A Figura 4.7, apresentada anteriormente, mostra que toda vez que um sensor de gás captura a variação de um gás, a persistência dos dados no banco de dados é invocada, e via *Framework* FProvW3C, são registradas as proveniências dos dados.

Os dados registrados deste aplicativo incluem informações de usuários do sistema, sensores, gases monitorados e agentes de software (55). Os dados registrados foram vinculados às ações dos usuários e agentes de software, auxiliando na rastreabilidade de cada ação executada. Por exemplo, a Figura 4.14(a), o sistema atribuiu o ID 36 a um usuário registrado. Então, usando a estrutura do FProvW3C, foi possível rastrear todos os dados gerados durante essa operação de registro, conforme Figura 4.14(b).

Como os dados vêm de várias fontes, é difícil identificar a origem de um problema. No entanto, como mostrado na Figura 4.14(b), a proveniência facilitou a avaliação deste sistema multiagente, permitindo-nos acompanhar todas as atividades que foram realizadas durante a sua execução.



4.14(a): Interface Smell App

ACTIVITY

ID	Description	Start Time	End Time
52	User 36has connected a Gases Device on port /dev/tty.usbmodem1411	2017-03-02 17:07:23.0	2017-03-02 17:07:23.0
53	The Gases Device Agent connected on port /dev/tty.usbmodem1411 measured environmental gases 10 times	2017-03-02 17:07:23.0	2017-03-02 17:07:29.0
54	The Gases Device Agent connected on port /dev/tty.usbmodem1411 measured gases from User36 28 times	2017-03-02 17:07:29.0	2017-03-02 17:07:46.0
55	Analyzer Agent calculated the percentage of change from environmental gases to the gases exhaled by the User: 36: 1. Methane: 3.27% 2. Hydrogen: 2.29% 3. Alcohol: 0.78% 4. CO2: 7.93 %	2017-03-02 17:07:46.0	2017-03-02 17:07:46.0
56	Alert Agent sent an alert to User 36 to seek a medical assistance: methane > hydrogen. (Report Id28)	2017-03-02 17:08:52.0	2017-03-02 17:08:52.0
57	Alert Agent verified data from all patients to generate alerts.	2017-03-02 17:08:52.0	2017-03-02 17:08:52.0

4.14(b): Proveniência no Smell App

Figura 4.14: Interceptação dos dados e Integração da captura ao SMA

Os outros cenários que foram avaliados o Prov-BDI4JADE, são os dos próprios exemplos disponibilizados pelo BDI4JADE. Por meio de uma interface de execução, os exemplos são iniciados juntamente com os objetivos que fazem com que os agentes sejam executados. Os exemplos apresentados pelo

BDI4JADE são:

- Agente *Hello World*: um aplicativo básico que retorna um *Hello World*.
- Agente *Hello World* com anotações de capacidade: o mesmo aplicativo *Hello World*, só que implementado usando anotações.
- Agentes de pingue-pongue: um aplicativo que mostra como trocar mensagens entre dois agentes pela plataforma.
- Agentes de metas compostas: um aplicativo que mostra como usar metas sequenciais e paralelas.
- Agente de falha de plano: mostra como diferentes planos são executados para atingir uma meta, quando há planos que falham.
- Agente de objetivo secundário: ilustra uma hierarquia de objetivos para atingir um objetivo de nível superior por um agente.
- Agente de capacidade múltipla: exemplo de uma aplicação que mostra como usar relacionamentos de capacidade.
- Mundo de blocos: a aplicação amplamente conhecida de blocos móveis para atingir um objetivo específico (blocos empilhados de uma maneira específica). Este exemplo mostra como usar objetivos declarativos.

Além dos cenários supracitados, que foram utilizados o Prov-BDI4JADE, outro cenário que podemos demostrar a importância da proveniência de dados e, permitir compreender as ações de sistemas autônomos é o apresentado em (53), onde a proveniência de dados pode ser utilizada para explicar ações que foram tomadas neste cenário e explicar acontecimentos como os de (25).

No cenário de (53), temos um exemplo extraído do Código Brasileiro de Trânsito (CBT), onde em um cruzamento sem semáforo, um conjunto de carros autônomos deve definir de quem é a prioridade. Como os agentes de software são construídos sobre normas, devem respeitar o CBT e neste caso, existe um conflito entre os artigos 29 e 38 do CBT, deixando a resolução deste conflito a cargo dos motoristas.

O conflito aqui é determinar quem tem a prioridade. Neste caso, o uso da proveniência pode explicar as interações que ocorreram entre cada agente e em caso de um acidente, permitir rastrear as ações de cada agente de software para determinar as causas do problema. Esse cenário, pode ser visto na Figura 1.1.

Além destes cenário, outros como os apresentados em (56, 57, 58, 59, 60), podem se beneficiar com a proveniência de dados, como mecanismo de rastrear e explicar as ações executadas nesses sistemas autônomos.

5
Avaliação Experimental

As pesquisas na área computacional, em geral, convergem para construção de software, modelos ou algoritmos (93). Os resultados obtidos nas respectivas pesquisas são validados através de avaliações, as quais estão cada vez mais rigorosas dentro da Engenharia de Software Experimental (ESE) (98). Como alternativa para auxílio à tarefa de comprovação dos resultados, positivos ou negativos, resultantes do estudo, apresenta-se o uso de métodos estatísticos.

A Estatística é aplicada em diversas áreas do conhecimento, fornecendo métodos para coleta, organização, análise e interpretação dos dados (95). O poder estatístico é uma parte inerente de estudos experimentais, os quais empregam testes de significância, essenciais para o planejamento, interpretação e validade das conclusões de um estudo (96).

Segundo (94), o processo de experimentação é composto por quatro partes sendo, respectivamente, i) definição; ii) planejamento; iii) execução e; iv) análise do experimento. O objetivo desse estudo experimental é a coleta de dados em um ambiente controlado, afim de confirmar ou negar os ganhos com o uso do Prov-BDI4JADE.

A próxima seção (5.1) descreve como foi realizado esse experimento.

5.1
Metodologia e Organização dos Experimentos

Conforme (65), estudo de caso é um método padrão usado para estudos experimentais em diversas ciências, incluindo a da computação, principalmente com a ESE. A vantagem dos estudos de caso é que eles são mais fáceis de planejar e são mais realistas, porém, como desvantagem, tem-se que os resultados são mais difíceis de generalizar e interpretar (65).

O estudo de caso conduzido nessa pesquisa, foi especificado para avaliar o Prov-BDI4JADE junto ao FProvW3C, onde verificou-se as vantagens e desvantagens do uso de proveniência de dados frente aos *Logs*, para tornar as ações de um sistema autônomo mais fáceis de serem compreendidas e explicativas.

Para a realização do estudo de caso, foram utilizados 3 grupos de participantes, que serão descriminados a seguir, todos voluntários e que estavam de

acordo com os termos da pesquisa, descritos no Anexo A.

Conforme (65), as entrevistas e questionários podem ser divididas em não-estruturadas, semiestruturadas e totalmente estruturadas. Um comparação entre esses tipos podem ser vistos na Tabela 5.1.

Tabela 5.1: Tipos de estudos.

	Não-estruturadas	Semiestruturadas	Totalmente estruturadas
Foco	Análise qualitativa a respeito da experiência dos indivíduos em relação a um fenômeno	Análise qualitativa e quantitativa a respeito da experiência dos indivíduos relação a um fenômeno	Pesquisadores buscam encontrar relações entre dois fenômenos.
Questões	Um roteiro de entrevista englobando as áreas especificadas no foco da pesquisa	Mistura entre questões abertas e fechadas	Questões fechadas
Objetivos	Exploratória	Descritiva e Exploratória	Descritiva e Exploratória

Com base na classificação de (65), esse estudo enquadra-se como semiestruturado, onde a primeira parte do estudo foi a caracterização dos indivíduos, seguindo o formulário disponível no Anexo B. De antemão, todos eram homens e graduandos nos cursos de Bacharelado em Sistemas de Informação ou de Engenharia de Software, do Centro de Ensino Superior de Juiz de Fora (CES/JF). Os resultados da caracterização dos indivíduos serão apresentados na seção 5.2.

Cada um dos 3 grupos analisaram os mesmos dados disponibilizados de forma diferente. Todos os dados foram gerados a partir da execução dos exemplos disponibilizados pelo BDI4JADE, conformes descritos em 4.4. O primeiro grupo analisou por meio de *Logs*, o segundo grupo utilizando os dados de proveniência, gerados pelo Prov-BDI4JADE e o terceiro grupo, analisou utilizando os *Logs* e os dados de proveniência, sendo uma junção dos anteriores.

Para conhecimento, os *Logs* utilizados para análise, encontram-se em https://tassio.eti.br/errors.log, já os dados de proveniência devem ser consultados via uma instância do *MongoDB* [1], e encontram-se disponíveis para download em https://tassio.eti.br/Prov-BDI4JADE.zip.

Utilizando-se destes dados, à cada grupo foi solicitado que respondesse os formulários disponíveis no Anexo C e no D. O primeiro formulário

[1]Disponível para download em: https://www.mongodb.com/

(Anexo C), era composto de questões objetivas, cuja questão era verificar se os participantes conseguiriam responder as perguntas a partir dos dados disponibilizados. Já o formulário do Anexo D, trata-se de uma avaliação qualitativa, sobre a opinião dos participantes com relação ao conjunto de dados disponibilizados e como classificam o Prov-BDI4JADE.

Todos os resultados destes questionários encontram-se na seção 5.2 a seguir.

5.2 Resultados Experimentais

Começando pela análise dos perfis dos participantes, fomentada pelos dados adquiridos pelo formulário do Anexo B, resaltando que em momento algum os participantes tiveram nenhum tipo de treinamento, temos o total de indivíduos por experimento conforme a Figura 5.1, a experiência de cada um conforme a Figura 5.2 e na sequência uma análise descritiva da idade dos indivíduos (Figura 5.3).

Nas Figuras 5.4 e 5.5 temos respectivamente uma análise descritiva da experiência acadêmica e da experiência profissional dos participantes. Já a Figura 5.6, temos a experiência com desenvolvimento de software. Por fim são analisadas a quantidade de participantes que conhecem sobre proveniência de dados (Figura 5.7), sobre sistemas multiagentes (Figura 5.8) e que utilizam sistemas de *Logs* em suas aplicações, conforme a Figura 5.9.

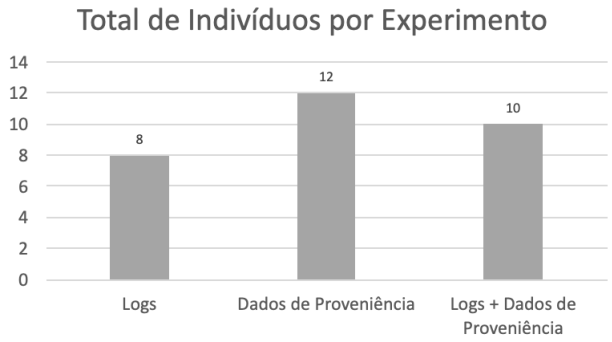
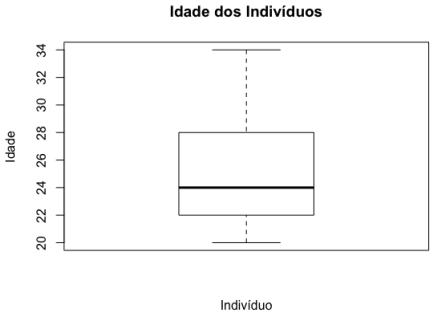


Figura 5.1: Indivíduos por Experimento

O total de participantes em cada grupo, deu-se por conta do número de presentes no dia de aplicação da avaliação, distribuídos de forma aleatória conforme o horário em que os mesmos podiam estar presente, dessa forma, conforme consta na Figura 5.1, alguns grupos tiveram mais participantes que outros. Um ponto a se destacar é na Figura 5.2, onde demostram que boa parte dos participantes já estão atuando no mercado e possuem experiência fora do ambiente acadêmico.

Figura 5.2: Experiência dos Indivíduos

Dos 30 participantes do experimento, conforme pode-se observar na Figura 5.3, apenas 1 participante não informou a idade, os demais possuem idade entre 20 e 34 anos, sendo a média 25,28 anos.




```
> summary(Perfil$Idade)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  20.00   22.00   24.00   25.28   28.00   34.00       1
```

Figura 5.3: Análise de Idade dos Indivíduos

Foi verificado também o tempo na academia, sendo que dos 30 participantes, 3 não informaram e dos demais, varia de 30 a 60 meses, sendo a média 43,37 meses, conforme Figura 5.4.

Já com relação à experiência profissional dos participantes, 10 indivíduos preferiram não informar, entre os demais, o intervalo varia de 1 mês a 192 meses, sendo a média 27,55 meses e média de 11 meses, conforme a Figura 5.5.

Na análise dos dados de experiência com desenvolvimento de software, observa-se que boa parte dos participantes, apesar de estarem atuando no mercado profissional, classificaram seus conhecimentos como teóricos, ainda sim, agregando os que se encontram em posição de estagiário, analista ou cargo superior, observa-se que são maioria os que possuem experiência com desenvolvimento, conforme a Figura 5.6.


```
> summary(Perfil$TA)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  34.00   35.00   45.00   43.37   48.00   60.00       3
```

Figura 5.4: Análise de Tempo na Academia em meses


```
> summary(Perfil$TI)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
   1.00    5.75   11.00   27.55   24.00  192.00      10
```

Figura 5.5: Análise de Tempo na Industria em meses

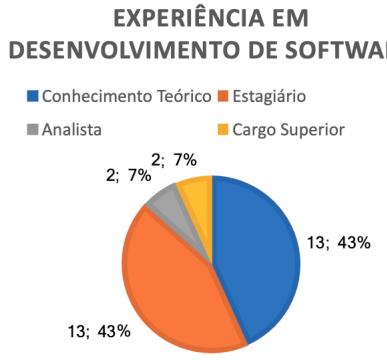


Figura 5.6: Experiência com Desenvolvimento

Quando perguntado sobre o conhecimento a respeito de proveniência de dados, apenas um participantes disso possuir (Figura 5.7) e sobre SMA, o número sob para 3 (Figura 5.8). Por fim, buscou-se verificar se os participantes fazem uso de sistemas de *Logs* em suas aplicações e 17 dos 30 participantes responderam que não, conforme a Figura 5.9.

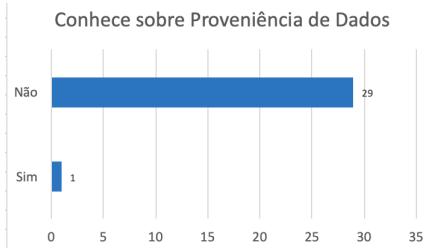


Figura 5.7: Conhecimento sobre proveniência

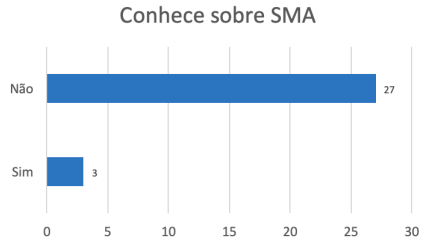


Figura 5.8: Conhecimento sobre SMA

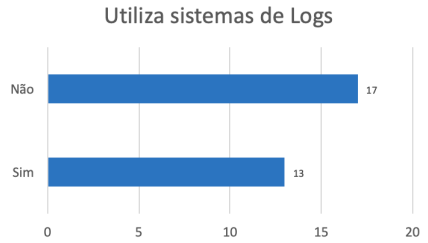
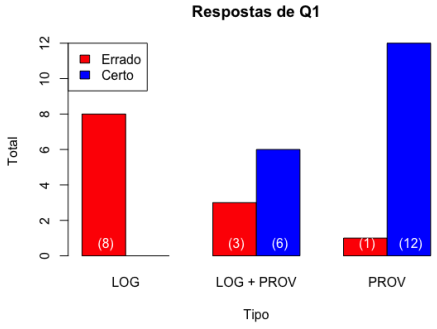


Figura 5.9: Conhecimento sobre *Logs*

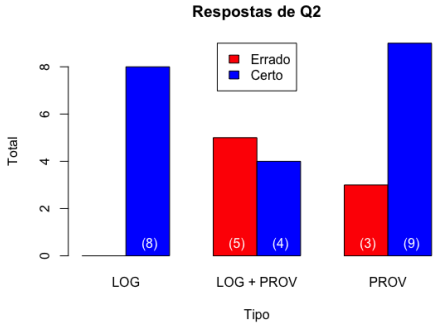
Essas informações podem auxiliar a explicar os resultados do experimento, uma vez que a experiência dos participantes sobre *Logs*, proveniência de dados e SMA, fazem parte do escopo deste trabalho e conforme (65), ajudam a caracterizar um estudo semiestruturado a qual cenário os resultados são válidos.

Passando para a análise dos dados objetivos, presentes no formulário do Anexo C, classificou-se cada uma das respostas entre "Certo" e "Errado", uma vez que, esperava-se dos participantes que por meio dos dados disponibilizados, encontrassem as respostas as questões, sendo os dados advindos da execução dos exemplos do BDI4JADE, apresentados na seção 4.4.

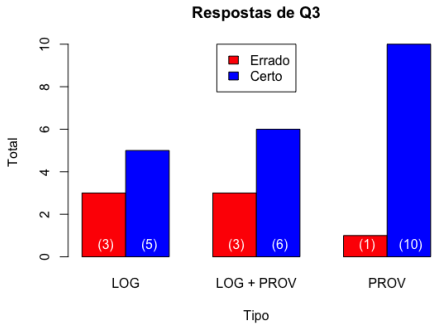
Como resultado, para cada grupo de participantes foi verificado a quantidade de acertos e erros, desse modo, podemos compará-los e verificar qual se saiu melhor, conforme a Figura 5.10. As questões de 1 a 10 são classificadas como fáceis, de 11 a 16 de nível médio e de 17 a 20 difíceis.



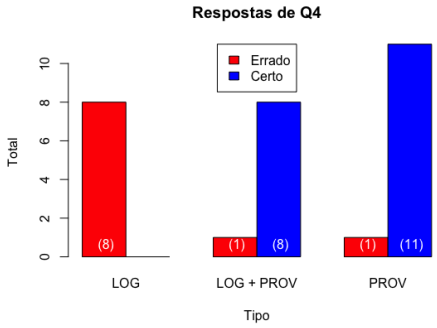
5.10(a): Resposta da Questão 1



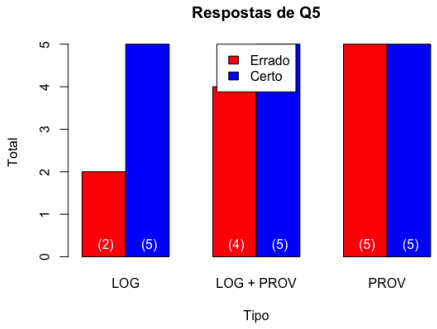
5.10(b): Resposta da Questão 2



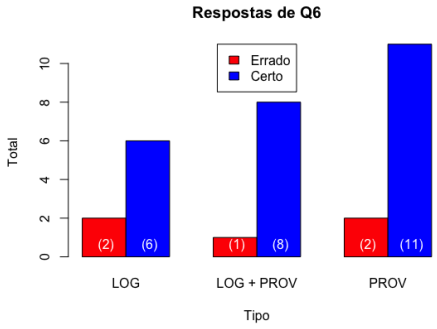
5.10(c): Resposta da Questão 3



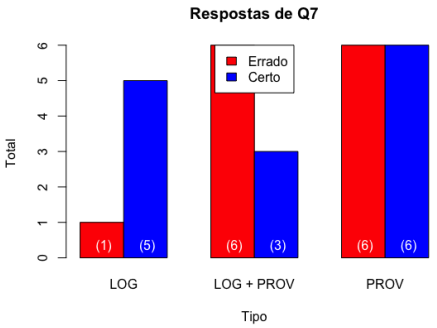
5.10(d): Resposta da Questão 4



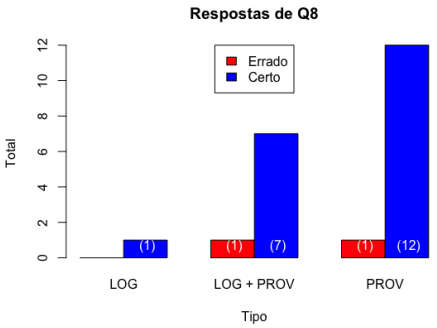
5.10(e): Resposta da Questão 5



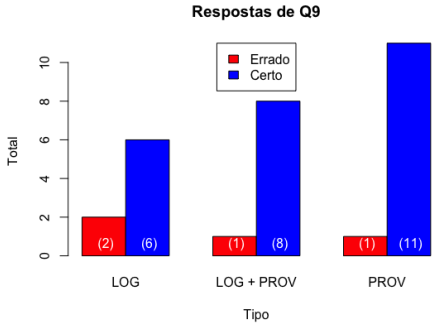
5.10(f): Resposta da Questão 6



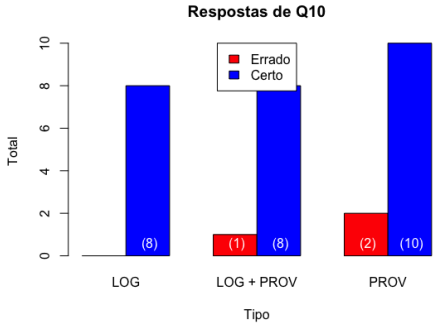
5.10(g): Resposta da Questão 7



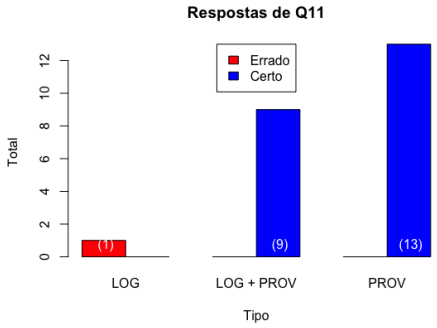
5.10(h): Resposta da Questão 8



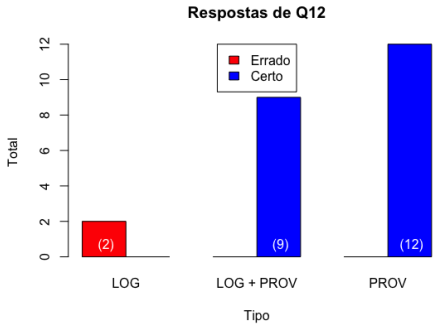
5.10(i): Resposta da Questão 9



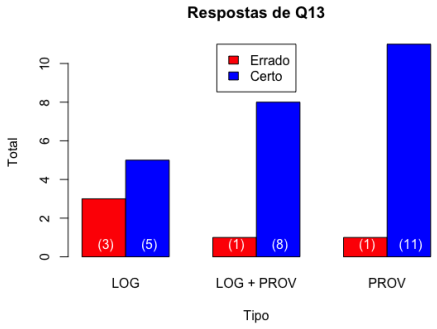
5.10(j): Resposta da Questão 10



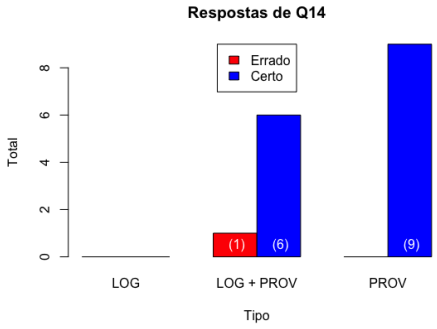
5.10(k): Resposta da Questão 11



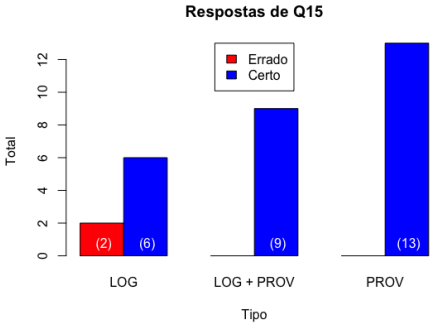
5.10(l): Resposta da Questão 12



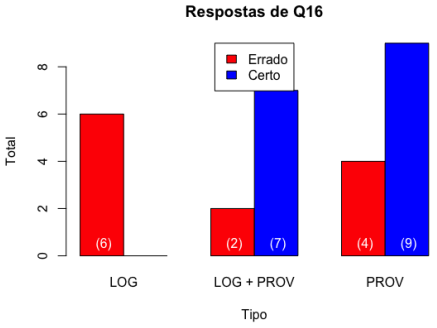
5.10(m): Resposta da Questão 13



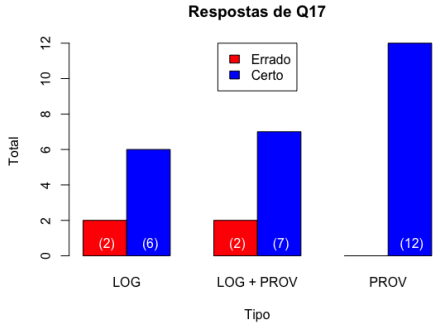
5.10(n): Resposta da Questão 14



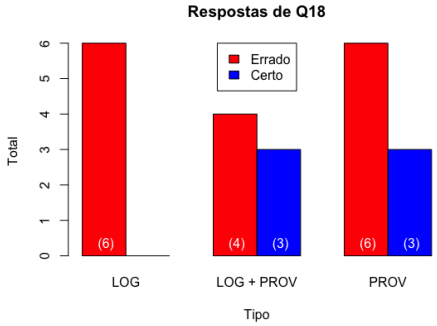
5.10(o): Resposta da Questão 15



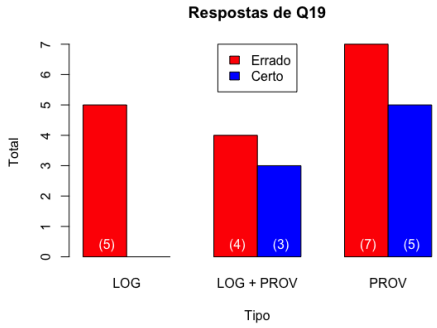
5.10(p): Resposta da Questão 16



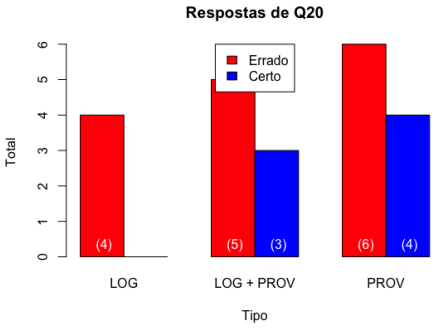
5.10(q): Resposta da Questão 17



5.10(r): Resposta da Questão 18



5.10(s): Resposta da Questão 19



5.10(t): Resposta da Questão 20

Figura 5.10: Comparativo entre as respostas do questionário objetivo.

Analisando o total de acertos e erros para cada questão, conforme apresentada pela Figura 5.10, temos que o grupo que utilizou os dados de proveniência coletados e armazenados pelo Prov-DBI4JADE, foi melhor em número de acertos do que o grupo que utilizou somente os *Logs*. Um detalhe é que o grupo que podia utilizar tanto os *Logs* como os dados de proveniência, foram intermediários entre o número de acertos e erros, isso se deve talvez a fonte de consulta para responder a questão. Até nas questões mais difíceis, em que o número de respostas erradas superou as de respostas certas, as respostas advindas dos dados de proveniência, foram melhores, tornando possível compreender o que o sistema executou.

Além da análise do número de acertos e erros por questão entre os grupos, foi verificado o tempo de resposta que cada grupo levou para cada questão. O resultado dessa análise pode ser visto na Tabela 5.2, onde podemos observar que entre os 3 grupos, o grupo mais rápido para responder foi o que analisou os *Logs*, seguido pelo grupo que analisou os *Logs* + Dados de proveniência, sendo o grupo que analisou somente por meio dos dados de proveniência, o mais demorado de todos.

Tabela 5.2: Tempo de Resposta para cada Questão

Questão	LOG				Dados de Proveniência				LOG + Dados de Proveniência			
	Min.	Máx.	Méd.	Desvio P.	Min.	Máx.	Méd.	Desvio P.	Min.	Máx.	Méd.	Desvio P.
Q1	2	12	5,375	3,248	1	2	2,615	2,022	1	11	4,000	3,807
Q2	1	4	1,625	1,187	1	8	2,923	1,934	1	4	1,778	0,971
Q3	1	9	3,500	2,672	1	7	2,923	1,891	1	3	1,556	0,881
Q4	0	5	2,000	1,927	1	8	2,846	2,115	1	4	1,889	0,927
Q5	0	10	2,750	3,058	0	10	4,692	3,591	2	6	3,000	1,414
Q6	1	3	1,625	0,744	1	10	2,923	2,396	1	2	1,667	0,500
Q7	1	4	1,625	1,302	0	7	2,923	2,215	2	6	4,222	1,481
Q8	0	4	0,500	1,414	1	5	2,615	1,192	0	3	1,667	0,866
Q9	1	3	1,875	0,640	0	9	3,308	2,428	1	3	2,222	0,666
Q10	1	4	2,000	0,925	0	10	2,462	2,569	1	2	1,556	0,527
Q11	0	3	0,375	1,060	2	7	3,769	1,423	1	5	2,889	1,269
Q12	2	12	5,375	3,248	1	9	2,615	2,022	1	11	4,000	3,807
Q13	1	3	1,875	0,834	0	4	2,308	1,109	1	3	1,778	0,666
Q14	-	-	-	-	0	8	3,846	2,511	2	6	3,778	1,201
Q15	1	2	1,125	0,353	1	4	2,462	0,8	1	2	1,333	0,500
Q16	0	10	2,250	3,240	1	11	2,462	2,665	1	2	1,444	0,527
Q17	0	6	2,500	1,690	1	5	2,846	1,214	1	3	1,778	0,666
Q18	0	4	2,000	1,511	0	9	3,231	2,554	0	6	3,222	1,855
Q19	0	9	2,250	2,964	0	5	2,846	1,463	1	3	1,778	0,833
Q20	0	2	0,625	0,744	0	10	3,321	2712	0	5	2,333	1,500
Média	0,632	5,737	2,171	1,724	0,600	7,400	2,997	137,506	1,000	4,500	2,395	1,243

Para verificar se a diferença entre os tempos de resposta entre os grupos eram significativas, os dados foram analisados por meio da estatística. Considerando um nível de significância de 5% para aceitação da hipótese nula, foi realizado o teste de *Shapiro-Wilk*, onde as hipóteses foram:

- H0 (hipótese nula): As amostras apresentam distribuição normal.
- H1 (hipótese alternativa): As amostras não apresentam distribuição normal.

O Teste de *Shapiro-Wilk* é utilizado para calcular o valor *W*, o qual refere-se à avaliação de uma amostra *Xj* quanto a distribuição normal. Em geral, o respectivo teste é utilizado para pequenos conjuntos de dados.

Como resultado do teste, todos os dados não apresentaram distribuição normal, rejeitando H0. Um histograma com a distribuição dos dados encontram-se disponível no Anexo E.

Com base nesse resultado, passa-se para a próxima análise que é a execução do teste não paramétrico. Como os dados são de 3 grupos, utilizou-se o teste de *Kruskal-Wallis* (KW) onde os valores de cada questão foi comparado entre os 3 grupos. Para essa análise, considerou-se as seguintes hipóteses:

- H0 (hipótese nula): Não há diferença entre as médias.
- H1 (hipótese alternativa): Há diferenças entre as médias.

O Teste *Kruskal-Wallis* é uma alternativa não paramétrica para a análise de variância (ANOVA), o qual, assim como grande parte dos testes paramétricos, baseia-se na substituição dos valores por seus *rankings* no conjunto de todos os valores.

Mantendo o um nível de significância de 5%, nota-se, que para as questões 1, 5, 7, 15, 19 e 20, à não aceitação da hipótese nula, de que as médias são estatisticamente equivalentes. Todas as demais questões, nota-se que não há diferença entre as médias, aceitando H0. Os resultados dessa analise encontram-se disponível na Figura 5.11.

```
        Kruskal-Wallis rank sum test

data:  DADOS$`DURAÇÃO Q1` by DADOS$TIPO
Kruskal-Wallis chi-squared = 6.5808, df = 2, p-value = 0.03724
```

5.11(a): Teste de KW para Questão 1

```
        Kruskal-Wallis rank sum test

data:  DADOS$`DURAÇÃO Q2` by DADOS$TIPO
Kruskal-Wallis chi-squared = 5.6553, df = 2, p-value = 0.05915
```

5.11(b): Teste de KW para Questão 2

```
        Kruskal-Wallis rank sum test

data:  DADOS$`DURAÇÃO Q3` by DADOS$TIPO
Kruskal-Wallis chi-squared = 5.3816, df = 2, p-value = 0.06783
```

5.11(c): Teste de KW para Questão 3

```
        Kruskal-Wallis rank sum test

data:  DADOS$`DURAÇÃO Q4` by DADOS$TIPO
Kruskal-Wallis chi-squared = 4.9643, df = 2, p-value = 0.08356
```

5.11(d): Teste de KW para Questão 4

```
        Kruskal-Wallis rank sum test

data:  DADOS$`DURAÇÃO Q5` by DADOS$TIPO
Kruskal-Wallis chi-squared = 9.5571, df = 2, p-value = 0.008408
```

5.11(e): Teste de KW para Questão 5

```
        Kruskal-Wallis rank sum test

data:  DADOS$`DURAÇÃO Q6` by DADOS$TIPO
Kruskal-Wallis chi-squared = 4.2817, df = 2, p-value = 0.1176
```

5.11(f): Teste de KW para Questão 6

```
        Kruskal-Wallis rank sum test

data:  DADOS$`DURAÇÃO Q7` by DADOS$TIPO
Kruskal-Wallis chi-squared = 6.067, df = 2, p-value = 0.04815
```

5.11(g): Teste de KW para Questão 7

```
        Kruskal-Wallis rank sum test

data:  DADOS$`DURAÇÃO Q8` by DADOS$TIPO
Kruskal-Wallis chi-squared = 4.5851, df = 2, p-value = 0.101
```

5.11(h): Teste de KW para Questão 8

```
        Kruskal-Wallis rank sum test

data:  DADOS$`DURAÇÃO Q9` by DADOS$TIPO
Kruskal-Wallis chi-squared = 5.0824, df = 2, p-value = 0.07877
```

5.11(i): Teste de KW para Questão 9

```
        Kruskal-Wallis rank sum test

data:  DADOS$`DURAÇÃO Q10` by DADOS$TIPO
Kruskal-Wallis chi-squared = 1.4361, df = 2, p-value = 0.4877
```

5.11(j): Teste de KW para Questão 10

```
        Kruskal-Wallis rank sum test

data:  DADOS$`DURAÇÃO Q11` by DADOS$TIPO
Kruskal-Wallis chi-squared = 2.066, df = 2, p-value = 0.3559
```

5.11(k): Teste de KW para Questão 11

```
        Kruskal-Wallis rank sum test

data:  d12$`DURAÇÃO 12` by d12$TIPO
Kruskal-Wallis chi-squared = 0.1, df = 1, p-value = 0.7518
```

5.11(l): Teste de KW para Questão 12

```
        Kruskal-Wallis rank sum test

data:  DADOS$`DURAÇÃO Q13` by DADOS$TIPO
Kruskal-Wallis chi-squared = 3.9639, df = 2, p-value = 0.1378
```

5.11(m): Teste de KW para Questão 13

```
        Kruskal-Wallis rank sum test

data:  DADOS$`DURAÇÃO Q14` by DADOS$TIPO
Kruskal-Wallis chi-squared = 0.49508, df = 1, p-value = 0.4817
```

5.11(n): Teste de KW para Questão 14

```
        Kruskal-Wallis rank sum test

data:  DADOS$`DURAÇÃO Q15` by DADOS$TIPO
Kruskal-Wallis chi-squared = 12.192, df = 2, p-value = 0.002252
```

5.11(o): Teste de KW para Questão 15

```
        Kruskal-Wallis rank sum test

data:  DADOS$`DURAÇÃO Q16` by DADOS$TIPO
Kruskal-Wallis chi-squared = 1.6567, df = 2, p-value = 0.4368
```

5.11(p): Teste de KW para Questão 16

```
        Kruskal-Wallis rank sum test

data:  DADOS$`DURAÇÃO Q17` by DADOS$TIPO
Kruskal-Wallis chi-squared = 5.9027, df = 2, p-value = 0.05227
```

5.11(q): Teste de KW para Questão 17

```
        Kruskal-Wallis rank sum test

data:  DADOS$`DURAÇÃO Q18` by DADOS$TIPO
Kruskal-Wallis chi-squared = 1.7027, df = 2, p-value = 0.4268
```

5.11(r): Teste de KW para Questão 18

```
        Kruskal-Wallis rank sum test

data:  DADOS$`DURAÇÃO Q19` by DADOS$TIPO
Kruskal-Wallis chi-squared = 6.1016, df = 2, p-value = 0.04732
```

5.11(s): Teste de KW para Questão 19

```
        Kruskal-Wallis rank sum test

data:  DADOS$`DURAÇÃO Q20` by DADOS$TIPO
Kruskal-Wallis chi-squared = 7.5323, df = 2, p-value = 0.02314
```

5.11(t): Teste de KW para Questão 20

Figura 5.11: Testes de Kruskal-Wallis para os tempos de resposta.

Como para as questões 1, 5, 7, 15, 19 e 20 o resultado foi a rejeição de H0, sendo necessário aplicar o teste de *Mann-Whitney-Wilcoxon* (MWW) para comparação 2 a 2 entre os grupos, essa segunda análise demostra se existe efetivamente uma diferença significativa entre as médias considerando um nível

de significância de 5%.

O Teste de *Mann-Whitney-Wilcoxon* refere-se a uma alternativa, não paramétrica, para o Teste T. O respectivo teste tem como requisito a necessidade de que as amostras sejam independentes, com dados contínuos e nas escalas ordinal, intervalar ou razão.

Como resultado para a questão 1, conforme a Figura 5.12, temos que entre o grupo que analisou usando somente *Logs* em comparação com quem utilizou os *Logs* + dados de proveniência, não existe diferença a um nível de significância de 5%, aceitando H0, o mesmo vale se observado entre o grupo *Logs* + dados de proveniência e o grupo proveniência. Já entre o grupo que utulizou *Logs* em relação ao de proveniência, existe diferença significativa entre as médias, levando então a aceitação da hipótese alternativa (H1). Nesse caso, olhando pela Tabela 5.2, para a Questão 1, a utilização dos dados proveniência para responder a questão se saiu melhor que os demais casos, uma vez que existe diferença significativa entre as médias.

```
        Pairwise comparisons using Wilcoxon rank sum test

data:  DADOS$`DURAÇÃO Q1` and DADOS$TIPO

           LOG    LOG + PROV
LOG + PROV 0.1442 -
PROV       0.0086 0.5724

P value adjustment method: none
> multcompLetters(TValor,compare = "<",threshold = 0.05, Letters = letters)
       LOG LOG + PROV       PROV
       "a"       "ab"        "b"
```

Figura 5.12: Teste de *MWW* para Questão 1.

Já para a questão 5, conforme a Figura 5.13, temos que entre o grupo que analisou usando somente *Logs* em comparação com quem utilizou os *Logs* + dados de proveniência, não existe diferença a um nível de significância de 5%, aceitando H0, entretanto, existe diferença significativa entre as médias com relação ao grupo que usou os dados de proveniência, levando então a rejeição da hipótese nula (H0).

Analisando a questão 7, conforme a Figura 5.14, temos que entre o grupo que analisou usando somente *Logs* em comparação com quem utilizou os *Logs* + dados de proveniência, existe diferença a um nível de significância de 5%, rejeitando H0, entretanto, se observado entre o grupo *Logs* + dados de proveniência e o grupo proveniência e entre o grupo dos *Logs* e o grupo de proveniência, temos a aceitação de H0. Ou seja, existe diferença significativa entre as médias, levando a rejeição da hipótese nula (H0).

A questão 15, conforme a Figura 5.15, comparando o grupo que utilizou somente *Logs* com quem utilizou os *Logs* + dados de proveniência, não existe

```
        Pairwise comparisons using Wilcoxon rank sum test

data:  DADOS$`DURAÇÃO Q5` and DADOS$TIPO

           LOG    LOG + PROV
LOG + PROV 0.4475 -
PROV       0.0174 0.0089

P value adjustment method: none
> multcompLetters(TValor,compare = "<",threshold = 0.05, Letters = letters)
       LOG LOG + PROV       PROV
       "a"         "a"        "b"
```

Figura 5.13: Teste de *MWW* para Questão 5.

```
        Pairwise comparisons using Wilcoxon rank sum test

data:  DADOS$`DURAÇÃO Q7` and DADOS$TIPO

           LOG   LOG + PROV
LOG + PROV 0.013 -
PROV       0.422 0.128

P value adjustment method: none
> multcompLetters(TValor,compare = "<",threshold = 0.05, Letters = letters)
       LOG LOG + PROV       PROV
       "a"         "b"       "ab"
```

Figura 5.14: Teste de *MWW* para Questão 7.

diferença a um nível de significância de 5%, aceitando H0, entretanto, se observado com o grupo de proveniência, temos a rejeição de H0. Ou seja, existe diferença significativa entre as médias, levando a rejeição da hipótese nula (H0).

```
        Pairwise comparisons using Wilcoxon rank sum test

data:  DADOS$`DURAÇÃO Q15` and DADOS$TIPO

           LOG    LOG + PROV
LOG + PROV 0.3601 -
PROV       0.0024 0.0126

P value adjustment method: none
> multcompLetters(TValor,compare = "<",threshold = 0.05, Letters = letters)
       LOG LOG + PROV       PROV
       "a"         "a"        "b"
```

Figura 5.15: Teste de *MWW* para Questão 15.

Conforme a Figura 5.16, para a questão 19, comparando o grupo que utilizou somente *Logs* com quem utilizou os *Logs* + dados de proveniência, não existe diferença a um nível de significância de 5%, aceitando H0, o mesmo é válido entre o grupo que utilizou os *Logs* com o grupo que utilizou os dados de proveniência, entretanto, se observado o grupo que utilizou a proveniência, com o grupo que utilizou os *Logs* + dados de proveniência, temos a rejeição de H0. Ou seja, existe diferença significativa entre as médias, levando a aceitação

da hipótese alternativa (H1).

```
        Pairwise comparisons using Wilcoxon rank sum test

data:  DADOS$`DURAÇÃO Q19` and DADOS$TIPO

           LOG   LOG + PROV
LOG + PROV 0.138 -
PROV       0.662 0.019

P value adjustment method: none
> multcompLetters(TValor,compare = "<",threshold = 0.05, Letters = letters)
       LOG LOG + PROV       PROV
      "ab"        "a"        "b"
```

Figura 5.16: Teste de MWW para Questão 19.

Por fim, conforme a Figura 5.17, para a questão 20, comparando o grupo que utilizou os *Logs* com o grupo que utilizou os dados de proveniência, não existe diferença a um nível de significância de 5%, aceitando H0, entretanto, se observado esses dois com o grupo que utilizou os *Logs*, temos a rejeição de H0. Ou seja, existe diferença significativa entre as médias, levando a aceitação da hipótese alternativa (H1).

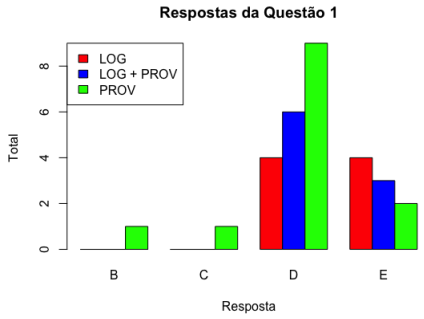
```
        Pairwise comparisons using Wilcoxon rank sum test

data:  DADOS$`DURAÇÃO Q20` and DADOS$TIPO

           LOG    LOG + PROV
LOG + PROV 0.0483 -
PROV       0.0087 0.5329

P value adjustment method: none
> multcompLetters(TValor,compare = "<",threshold = 0.05, Letters = letters)
       LOG LOG + PROV       PROV
       "a"        "b"        "b"
```

Figura 5.17: Teste de MWW para Questão 20.

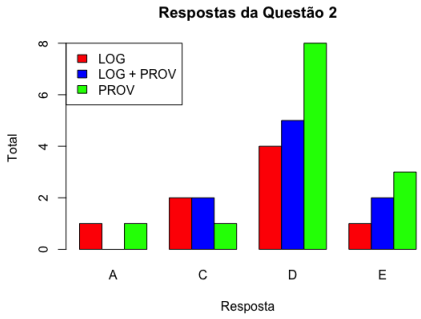
Com base nessa análise, temos que para a maior parte das questões não existe diferença de tempo significativa para serem respondidas, e associando com a análise anterior de acertos e erros, temos que quem utilizou os dados de proveniência conseguiu acertar mais do que quem utilizou somente os *Logs*, demostrando que o uso da proveniência tornou o sistema autônomo mais explicativo sobre suas ações.

Além dessa análise do questionário objetivo, temos no Anexo D, o formulário qualitativo, que foi aplicado aos participantes. As respostas das questões de 1 à 5 podem ser vistas na Figura 5.18.

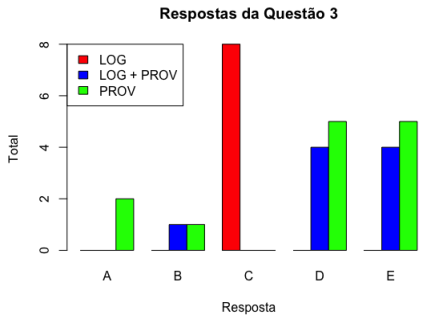
Essas respostas auxiliam a fomentar a hipótese geral, apresentada na seção 1.3, onde podemos considerar que o uso da proveniência auxilia a explicar as ações realizadas por um sistema autônomo, tonando-o mais explicativos.



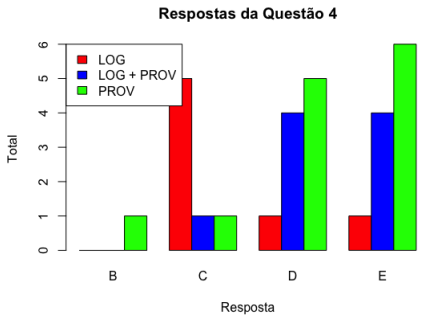
5.18(a): Respostas da Questão 1



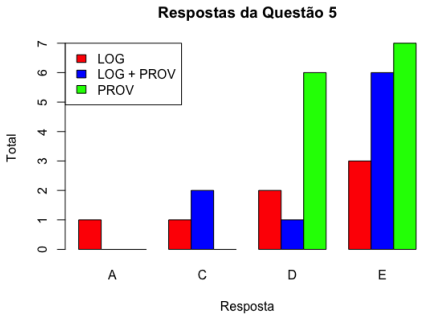
5.18(b): Respostas da Questão 2



5.18(c): Respostas da Questão 3



5.18(d): Respostas da Questão 4



5.18(e): Respostas da Questão 5

Figura 5.18: Respostas do questionário qualitativo.

Além disso, o tempo de resposta entre os grupos para a maior parte das questões, não apresentou diferença significativa no tempo para responder, e por fim, os participantes que utilizaram os dados de proveniência, agruparam-se entre as respostas "Concordo parcialmente" e "Concordo Totalmente" sobre as questões que perguntavam sobre a utilidade dos dados fornecidos, conforme a Figura 5.18.

As questões de 6 à 10, presentes no formulário do Anexo D, visam validar as hipóteses especificas apresentadas na seção 1.3, onde com base nas respostas dos participantes, sintetizando, temos para a questão 6, que os mesmos consideram importante pois se tratando de um sistema autônomo e seu

comportamento deve ser compreensível, para a questão 7, no geral, afirmou-se que não tinham conhecimento sobre quais outras informações poderiam ser úteis. Já para a questão 8, justificaram que um *Framework* torna mais fácil o processo de implementação, para a questão 9, apontaram a grande quantidade de informações geradas de forma detalhada pelo Prov-BDI4JADE.

Por fim, para a questão 10, que por conta da quantidade de informações geradas, a descoberta e visualização dos dados deveria ser melhorada.

Com as análises estatísticas realizadas tendo como base a avaliação quantitativa e qualitativa, podemos considerar a aceitação da Hipótese geral nula, onde o uso de proveniência de dados auxilia na análise das ações realizadas por sistemas autônomos, tornando-os explicativos. Para a Hipótese específica sobre o *Framework* FProvW3C, também podemos considerar H0, onde o uso do *framework* de software FProvW3C, para coleta de dados de proveniência em sistemas autônomos, possibilita torná-los explicativos, levando em consideração os resultados da avaliação quantitativa.

E por fim, com base nos resultados qualitativos sobre o Prov-BDi4JADE, podemos considerar a aceitação de H0 (Hipótese nula), em que o uso da plataforma Prov-BDI4JADE possibilita abstrair os detalhes da configuração de coleta dos dados de proveniência, tornando o sistemas autônomos explicativos, sem demandar conhecimento sobre os modelos de proveniência.

Essas respostas dadas pelos participantes, justificam que o FProvW3C facilita o trabalho de coleta e armazenamento das informações seguindo um modelo de proveniência e o Prov-BDI4JADE dispensa do programador conhecer um modelo de proveniência e que as informações geradas auxiliam a responder questões sobre as ações executadas por um sistema autônomo.

Ressalta-se mais uma vez que os participantes não receberam nenhum treinamento e que por conta disso os resultados são validos para esse cenário, uma vez que se os mesmos conhecessem mais sobre sistemas multagentes ou do modelo de proveniência os resultados poderiam ser melhorados.

6
Considerações Finais

Neste trabalho foram abordadas as arquiteturas de agentes e sistemas multiagentes existentes, sendo elas implementações de sistemas autônomos, bem como algumas das plataformas mais conhecidas de agentes. Discutiu-se sobre o desenvolvimento e sobre os softwares que fazem uso de inteligência artificial distribuída e como podemos compreender melhor a série de ações executadas por um sistema autônomo, e sua importância de serem explicáveis. Neste ponto foi abordado o registro histórico do sistema com uso de *Logs* e por meio de proveniência, abordando as vantagens e limitações de cada um.

O uso de proveniência de dados pela informática vem crescendo e sua aplicação em sistemas autônomos possibilita compreender ações e resultados até então desconhecidos; ligando as linhas de engenharia de software, banco de dados e inteligência artificial. A engenharia de software experimental é outra vertente que pode se beneficiar com a proveniência, visto que para avaliar software de modo empírico deve-se observar o máximo de detalhes possíveis, sendo este o objetivo do uso da proveniência.

A proveniência de dados possibilita registrar informações em diferentes níveis de abstração, e seu uso por meio de aspectos é uma possibilidade para que os dados sejam capturados, sem que haja a necessidade de inserção direta de código na arquitetura da plataforma multiagente. O uso de aspectos se posiciona como uma abordagem viável para este caso, contudo, deve-se ter a consciência das consequências dessa abordagem.

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 abordado no trabalho. Em sistemas multiagentes há muitas oportunidades para aplicar a proveniência dos dados.

Atualmente algumas plataformas apresentam o uso de *Logs*, de maneira simples e que não permitem uma clara compreensão da execução do sistema em softwares complexos. A proveniência de dados é uma solução emergente, que ainda não foi abordada pela comunidade e que neste trabalho pode ser considerado um ponta pé inicial.

Esse trabalho conforme abordado em (46), é o passo inicial da discussão sobre o uso de proveniência de dados em sistemas multiagentes e como isto deve ser feito. Um mecanismo único de captura para todas as plataformas é utópico, visto a diferença entre as mesma e, a inserção de código direto na plataforma pode ser ainda pior, pois torna a manutenção da captura complexa, cresce o sistema em número de classes, métodos, números de linhas e dependendo do caso, em complexidade.

6.1 Contribuições do Trabalho

Esta pesquisa buscou pelos melhores resultados sobre o modo de captura do dados em sistemas autônomos, de forma não invasiva. Os dados de um sistema multiagente são valiosos para entender a dinâmica do sistema, e tanto a captura quanto o armazenamento são atividades críticas que devem ser bem planejadas e executadas para não invalidar as ações do sistema.

A proveniência de dados permite rastrear a origem dos dados e os processos de derivação que ocorreram entre a origem dos dados e o estado em que os dados são encontrados em um determinado momento. Se considerarmos que o modelo de proveniência contribui para a avaliação da qualidade dos dados e, consequentemente, do processo que os gerou, ele suporta maior confiança nas ações tomadas pelos agentes dentro de um sistema autônomo.

Com isso, temos que a originalidade e a relevância desse trabalho estão focados nas contribuições desta pesquisa, que são:

- Uma comparação entre os principais modelos de proveniência e qual melhor se aplica a sistemas autônomos;
- Desenvolvimento do *Framework* FProvW3C;
- Formalização e aplicação de um modelo de proveniência para sistemas autônomos (Prov-BDI4JADE);
- Tornar o comportamento de sistemas autônomos explicáveis por meio de proveniência.

Vale ressaltar que ao usar a proveniência para capturar os dados em um sistema autônomo, será possível avançar em outros campos, que podem envolver: i) semântica e sintaxe de dados, ii) ontologia, iii) mineração de dados e iv) dados em outros formatos, como *XML* ou *JSON*.

Além disso, outras contribuições dessa pesquisa são os trabalhos publicados ao longo dessa tese, sendo os artigos (26), (44), (46), (53), (59), (61), (99), (100), (101), (102), (103), frutos desse estudo.

6.2 Limitações e Trabalhos Futuros

A proposta dessa tese foi utilizar a proveniência de dados capturada por meio de aspectos para tornar o comportamento de sistemas autônomos explicáveis. Como todo trabalho, está seção tem por objetivo definir alguns pontos sobre essa proposta, tanto positivos como negativos que podem de alguma forma serem tratados como limitação.

Começando pelos pontos positivos em utilizar a proveniência de dados para explicar o comportamento de sistemas multiagentes, apoiada em uma abordagem de aspectos para este caso específico, temos:

- Não afeta a arquitetura da plataforma de sistema multiagente;
- Permite interceptar qualquer ponto de execução identificável do sistema;
- Falha no processo de captura da proveniência não afeta o software;
- Permitir vincular os eventos capturados usando o modelo PROV, por meio do *Framework* FProvW3C;
- Permite o reuso do código de captura em qualquer software que utilize a mesma base de plataforma de sistema multiagente (JADE, DBI4JADE);
- Facilidade para manutenção;
- Fácil acoplamento com o sistema multiagente;

Contudo, também devemos destacar alguns pontos negativos com o uso da abordagem com aspecto, que podem ser:

- Garantia que o ponto de corte está ocorrendo no local certo;
- Garantia que o dado capturado está correto;
- Excesso de chamadas na captura da proveniência em uma função;
- Código complexo após a integração da plataforma com os aspectos usados para a captura de proveniência;

Assumir que o uso de aspectos é a melhor opção para a captura da proveniência seria ingênuo ou imaturo, visto que depende do acesso ao código da plataforma e não foi discutida outas abordagens para essa captura neste trabalho além do uso de *Logs*. Contudo, essa se mostra uma solução viável, dado que se trata de um interesse transversal, conforme supramencionado.

Como trabalhos futuros, pretende-se melhorar a forma de visualização e filtragem dos dados, uma vez que esse foi o ponto mais questionado pelos participantes do experimento, e consequentemente aplicar outros mecanismos sobre os dados, tais como ontologia e máquinas de inferência.

Referências bibliográficas

1 GUBBI, J.; BUYYA, R.; MARUSIC, S. ; PALANISWAMI, M.. **Internet of things (iot): A vision, architectural elements, and future directions**. Future generation computer systems, 29(7):1645–1660, 2013.

2 JENNINGS, N. R.; WOOLDRIDGE, M.. **Agent-oriented software engineering**. Handbook of agent technology, 18, 2001.

3 JENNINGS, N. R.. **An agent-based approach for building complex software systems**. Communications of the ACM, 44(4):35–41, 2001.

4 YU, H.; SHEN, Z. ; LEUNG, C.. **From internet of things to internet of agents**. In: GREEN COMPUTING AND COMMUNICATIONS (GREENCOM), 2013 IEEE AND INTERNET OF THINGS (ITHINGS/CPSCOM), IEEE INTERNATIONAL CONFERENCE ON AND IEEE CYBER, PHYSICAL AND SOCIAL COMPUTING, p. 1054–1057. IEEE, 2013.

5 ZAMBONELLI, F.; JENNINGS, N. R.; OMICINI, A. ; WOOLDRIDGE, M. J.. **Agent-oriented software engineering for internet applications**. In: COORDINATION OF INTERNET AGENTS, p. 326–346. Springer, 2001.

6 WOOLDRIDGE, M.; JENNINGS, N. R.. **Intelligent agents: Theory and practice**. The knowledge engineering review, 10(2):115–152, 1995.

7 JENNINGS, N. R.. **Coordination techniques for distributed artificial intelligence**. Foundations of distributed artificial intelligence, p. 187–210, 1996.

8 GIRARDI, R.. **Engenharia de software baseada em agentes**. In: PROCEDIMENTOS DO IV CONGRESSO BRASILEIRO DE CIÊNCIA DA COMPUTAÇÃO (CBCOMP 2004). sn, 2004.

9 WOOLDRIDGE, M.. **An introduction to multiagent systems**. John Wiley & Sons, 2009.

10 RAO, A. S.; GEORGEFF, M. P. ; OTHERS. **Bdi agents: from theory to practice.** In: ICMAS, volumen 95, p. 312–319, 1995.

11 SILVA, J. C.. **Um modelo para avaliação de aprendizagem no uso de ferramentas síncronas em ensino mediado pela Web**. PhD thesis, Tese de Doutorado, PUC-RIO, 2004.

12 FERREIRA, S. L. C.; GIRARDI, R.. **Arquiteturas de software baseadas em agentes: do nível global ao detalhado**. Revista Eletrônica de Iniciação Científica da SBC, 2002.

13 MILES, S.; GROTH, P.; MUNROE, S.; LUCK, M. ; MOREAU, L.. **Agentprime: Adapting mas designs to build confidence**. In: INTERNATIONAL WORKSHOP ON AGENT-ORIENTED SOFTWARE ENGINEERING, p. 31–43. Springer, 2007.

14 SHAW, P. H.; FARWER, B. ; BORDINI, R. H.. **Theoretical and experimental results on the goal-plan tree problem**. In: PROCEEDINGS OF THE 7TH INTERNATIONAL JOINT CONFERENCE ON AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS-VOLUME 3, p. 1379–1382. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

15 WINIKOFF, M.; CRANEFIELD, S.. **On the testability of bdi agent systems**. Journal of Artificial Intelligence Research, 51:71–131, 2014.

16 MILES, S.; MUNROE, S.; LUCK, M. ; MOREAU, L.. **Modelling the provenance of data in autonomous systems**. In: PROCEEDINGS OF THE 6TH INTERNATIONAL JOINT CONFERENCE ON AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS, p. 50. ACM, 2007.

17 DE ARAÚJO, T.; VON STAA, A.. **Supporting failure diagnosis with logs containing meta-information annotations**. Technical report, Technical Reports in Computer Science. PUC-Rio. ISSN 0103-9741, 2014.

18 GÜLCÜ, C.. **The complete log4j manual**. QOS. ch, 2003.

19 HOLZINGER, A.; BIEMANN, C.; PATTICHIS, C. S. ; KELL, D. B.. **What do we need to build explainable ai systems for the medical domain**. arXiv preprint arXiv:1712.09923, 2017.

20 LOPEZ, F. L.. **Social power and norms**. Diss. University of Southampton, 2003.

21 GUNNING, D.. **Explainable artificial intelligence (xai)**. Defense Advanced Research Projects Agency (DARPA), 2017.

22 CORE, M. G.; LANE, H. C.; VAN LENT, M.; GOMBOC, D.; SOLOMON, S. ; ROSENBERG, M.. **Building explainable artificial intelligence systems**. In: AAAI, p. 1766–1773, 2006.

23 GOEBEL, R.; CHANDER, A.; HOLZINGER, K.; LECUE, F.; AKATA, Z.; STUMPF, S.; KIESEBERG, P. ; HOLZINGER, A.. **Explainable ai: the new 42?** In: INTERNATIONAL CROSS-DOMAIN CONFERENCE FOR MACHINE LEARNING AND KNOWLEDGE EXTRACTION, p. 295–303. Springer, 2018.

24 COELHO, R.; KULESZA, U.; VON STAA, A. ; LUCENA, C.. **Unit testing in multi-agent systems using mock agents and aspects**. In: PROCEEDINGS OF THE 2006 INTERNATIONAL WORKSHOP ON SOFTWARE ENGINEERING FOR LARGE-SCALE MULTI-AGENT SYSTEMS, p. 83–90. ACM, 2006.

25 LEE, D.. **Sensor firm Velodyne 'baffled' by Uber selfdriving death**. <https://www.bbc.com/news/technology-43523286>, 2018. [Online; Acessado em: 05-Janeiro-2019].

26 SIRQUEIRA, T.; VIANA, M.; NASCIMENTO, N. ; LUCENA, C.. **A software framework for data provenance**. In: 29TH INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING & KNOWLEDGE ENGINEERING (SEKE'2017). SEKE/KNOWLEDGE SYSTEMS INSTITUTE, p. 615–619, 2017.

27 DALPRA, H. L.; COSTA, G. C. B.; SIRQUEIRA, T. F.; BRAGA, R. M.; CAMPOS, F.; WERNER, C. M. L. ; DAVID, J. M. N.. **Using ontology and data provenance to improve software processes.** In: ONTOBRAS, 2015.

28 SIMMHAN, Y. L.; PLALE, B. ; GANNON, D.. **A survey of data provenance techniques**. Computer Science Department, Indiana University, Bloomington IN, 47405:69, 2005.

29 FREIRE, J.; KOOP, D.; SANTOS, E. ; SILVA, C. T.. **Provenance for computational tasks: A survey**. Computing in Science & Engineering, 10(3), 2008.

30 HOLLAND, J. H.. **Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence**. MIT press, 1992.

31 RUSSELL, S. J.; NORVIG, P.. **Artificial intelligence: a modern approach**. Malaysia; Pearson Education Limited,, 2016.

32 SHAHRIARI, B.; SWERSKY, K.; WANG, Z.; ADAMS, R. P. ; DE FREITAS, N.. **Taking the human out of the loop: A review of bayesian optimization**. Proceedings of the IEEE, 104(1):148–175, 2016.

33 HUHNS, M. N.; STEPHENS, L. M.. **Multiagent systems and societies of agents**. Multiagent systems: a modern approach to distributed artificial intelligence, 1:79–114, 1999.

34 MALDONADO, J. C.; BRAGA, R. T. V.; GERMANO, F. S. R. ; MASIERO, P. C.. **Padrões e frameworks de software**. Notas Didáticas, Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo, ICMC/USP, São Paulo, SP, Brasil, 2002.

35 KRAVARI, K.; BASSILIADES, N.. **A survey of agent platforms**. Journal of Artificial Societies and Social Simulation, 18(1):11, 2015.

36 BELLIFEMINE, F.; POGGI, A. ; RIMASSA, G.. **Jade–a fipa-compliant agent framework**. In: PROCEEDINGS OF PAAM, volumen 99, p. 33. 1999.

37 O'BRIEN, P. D.; NICOL, R. C.. **Fipa—towards a standard for software agents**. BT Technology Journal, 16(3):51–59, 1998.

38 POKAHR, A.; BRAUBACH, L. ; LAMERSDORF, W.. **Jadex: A bdi reasoning engine**. In: MULTI-AGENT PROGRAMMING, p. 149–174. Springer, 2005.

39 HOWDEN, N.; RÖNNQUIST, R.; HODGSON, A. ; LUCAS, A.. **Jack intelligent agents-summary of an agent infrastructure**. In: 5TH INTERNATIONAL CONFERENCE ON AUTONOMOUS AGENTS, 2001.

40 BORDINI, R. H.; HÜBNER, J. F.. **Bdi agent programming in agentspeak using jason**. In: INTERNATIONAL WORKSHOP ON COMPUTATIONAL LOGIC IN MULTI-AGENT SYSTEMS, p. 143–164. Springer, 2005.

41 NUNES, I.; LUCENA, C. ; LUCK, M.. **Bdi4jade: a bdi layer on top of jade**. ProMAS 2011, p. 88–103, 2011.

42 MOREAU, L.; FREIRE, J.; FUTRELLE, J.; MCGRATH, R. E.; MYERS, J. ; PAULSON, P.. **The open provenance model: An overview**. In: INTERNATIONAL PROVENANCE AND ANNOTATION WORKSHOP, p. 323–326. Springer, 2008.

43 MISSIER, P.; BELHAJJAME, K. ; CHENEY, J.. **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, 2013.

44 DA CUNHA, F. J. P.; SIRQUEIRA, T. F. M.; VIANA, M. L. ; PEREIRA, C. J.. **Extending bdi multiagent systems with agent norms**. International Journal of Computer and Information Engineering, 12(5):302–309, 2018.

45 NETO, B. F. D. S.; DA SILVA, V. T. ; DE LUCENA, C. J. P.. **Nbdi: An architecture for goal-oriented normative agents.** In: ICAART (1), p. 116–125, 2011.

46 SIRQUEIRA, T. F. M.; VIANA, M. L.; CUNHA, F. J. P. D.; NUNES, I. ; LUCENA, C. J. P. D.. **Data provenance in multi-agent systems: relevance, benefits and research opportunities**. International Journal of Metadata, Semantics and Ontologies, 13(1):9–19, 2018.

47 PERIARD, G.. **O que é o 5w2h e como ele é utilizado?** <http://www.sobreadministracao.com/o-que-e-o-5w2h-e-como-ele-e-utilizado/>, 2009. [Online; Acessado em: 05-Janeiro-2019].

48 TAN, W. C.. **Research problems in data provenance.** IEEE Data Eng. Bull., 27(4):45–52, 2004.

49 GARCIA, A.; SANT'ANNA, C.; FIGUEIREDO, E.; KULESZA, U.; LUCENA, C. ; VON STAA, A.. **Modularizing design patterns with aspects: a quantitative study**. In: TRANSACTIONS ON ASPECT-ORIENTED SOFTWARE DEVELOPMENT I, p. 36–74. Springer, 2006.

50 VON STAA, A.. **Programação Modular: Desenvolvendo programas complexos de forma organizada e segura**. Campos, 2000.

51 LADDAD, R.. **Aspectj in action: enterprise AOP with spring applications**. Manning Publications Co., 2009.

52 CUNHA, F.; DA COSTA, A. D.; VIANA, M. ; DE LUCENA, C. J. P.. **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, volumen 2, p. 186–189. IEEE, 2015.

53 CUNHA, F.; VIANA, M.; SIRQUEIRA, T.; ROSEMBERG, M. ; LUCENA, C.. **Understanding normative bdi agents behavior**. In: SEKE, p. 221–224, 2018.

54 BALDONI, M.; BAROGLIO, C.; MASCARDI, V.; OMICINI, A. ; TORRONI, P.. **Agents, multi-agent systems and declarative programming: what, when, where, why, who, how?** In: A 25-YEAR PERSPECTIVE ON LOGIC PROGRAMMING, p. 204–230. Springer, 2010.

55 DO NASCIMENTO, N. M.; VIANA, M. L. ; DE LUCENA, C. J. P.. **An iot-based tool for human gas monitoring**. In: IXV CONGRESSO BRASILEIRO DE INFORMATICA EM SAUDE (CBIS), volumen 1, p. 96–98, 2016.

56 KIFOR, T.; VARGA, L. Z.; VAZQUEZ-SALCEDA, J.; ALVAREZ, S.; WILLMOTT, S.; MILES, S. ; MOREAU, L.. **Provenance in agent-mediated healthcare systems**. IEEE Intelligent Systems, 21(6):38–46, 2006.

57 NAJA, I.; MOREAU, L. ; ROGERS, A.. **Provenance of decisions in emergency response environments**. In: INTERNATIONAL PROVENANCE AND ANNOTATION WORKSHOP, p. 221–230. Springer, 2010.

58 KOHWALTER, T.; CLUA, E. ; MURTA, L.. **Provenance in games**. In: BRAZILIAN SYMPOSIUM ON GAMES AND DIGITAL ENTERTAINMENT (SBGAMES), p. 11, 2012.

59 SIRQUEIRA, T.; VIANA, M.; FERNANDES, C.; NASCIMENTO, N.; SASTRE, J.; MIRANDA, P. A.; AUGUSTO, V. ; LUCENA, C.. **Design de uma plataforma para a gestão de informações médicas distribuídas baseada em sistemas multiagentes e proveniência de dados**. Monografias em Ciência da Computação, 11, 2018.

60 MILES, S.; WONG, S. C.; FANG, W.; GROTH, P.; ZAUNER, K.-P. ; MOREAU, L.. **Provenance-based validation of e-science experiments**. Web Semantics: Science, Services and Agents on the World Wide Web, 5(1):28–38, 2007.

61 DA CUNHA, F. J. P.; SIRQUEIRA, T. F. M.; VIANA, M. L. ; PEREIRA, C. J.. **Extending bdi multiagent systems with agent norms**. In: INTERNATIONAL CONFERENCE ON INTELLIGENT AGENT TECHNOLOGY, 2018.

62 TEMPLETON, B.. **Tesla Autopilot Repeats Fatal Crash; Do They Learn From Past Mistakes?** <https://www.forbes.com/sites/bradtempleton/2019/05/21/tesla-autopilot-repeats-fatal-crash-do-they-learn-from-past-mistakes/#2d1dd44e2f2e>, 2019. [Online; Acessado em: 17-Agosto-2019].

63 GUARDIAN, T.. **Ethiopian Airlines crash: anti-stall system 'engaged repeatedly'**. <https://www.theguardian.com/business/2019/apr/03/ethiopian-airlines-crash-boeing-software-engaged-repeatedly>, 2019. [Online; Acessado em: 17-Agosto-2019].

64 BASILI, V. R.. **Applying the goal/question/metric paradigm in the experience factory**. Software Quality Assurance and Measurement: A Worldwide Perspective, 7(4):21–44, 1993.

65 WOHLIN, C.; RUNESON, P.; HÖST, M.; OHLSSON, M. C.; REGNELL, B. ; WESSLÉN, A.. **Experimentation in software engineering**. Springer Science & Business Media, 2012.

66 VON WANGENHEIM, C. G.; RUHE, G.. **Análise de custo e beneficio de mensuração baseada em gqm-um estudo de caso replicado**.

67 CHAVEZ, C.. **Um enfoque baseado em modelos para o design orientado a aspectos**. PhD thesis, Tese de Doutorado-Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, 2004.

68 GARCIA, A.. **Objetos e Agentes: Uma Abordagem Orientada a Aspectos. Departamento de Informática**. PhD thesis, Tese de Doutorado-Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, 2004.

69 FOUNDATION, F. S.. **GNU LESSER GENERAL PUBLIC LICENSE**. <https://www.gnu.org/licenses/lgpl-3.0.pt-br.html>, 2007. [Online; Acessado em: 21-Agosto-2019].

70 RAO, A. S.. **Agentspeak (l): Bdi agents speak out in a logical computable language**. In: EUROPEAN WORKSHOP ON MODELLING AUTONOMOUS AGENTS IN A MULTI-AGENT WORLD, p. 42–55. Springer, 1996.

71 ALVES, T. O. M.; OTHERS. **SciProvMiner: Captura e Consulta de Proveniência utilizando Recursos Web Semânticos para Ampliação do Conhecimento Gerado e Otimização do Processo de Coleta**. PhD thesis, Dissertação de Mestrado, PGCC/UFJF, Juiz de Fora, MG, Brasil, 2013.

72 PRABHUNE, A.; ZWEIG, A.; STOTZKA, R.; GERTZ, M. ; HESSER, J.. **Prov2one: an algorithm for automatically constructing provone provenance graphs**. In: INTERNATIONAL PROVENANCE AND ANNOTATION WORKSHOP, p. 204–208. Springer, 2016.

73 PRABHUNE, A.; ANSARI, H.; KESHAV, A.; STOTZKA, R.; GERTZ, M. ; HESSER, J.. **Metastore: A metadata framework for scientific data repositories**. In: 2016 IEEE INTERNATIONAL CONFERENCE ON BIG DATA (BIG DATA), p. 3026–3035. IEEE, 2016.

74 TRAVASSOS, G. H.; DOS SANTOS, P. S. M.; MIAN, P. G.; NETO, A. C. D. ; BIOLCHINI, J.. **An environment to support large scale experimentation in software engineering**. In: 13TH IEEE INTERNATIONAL CONFERENCE

ON ENGINEERING OF COMPLEX COMPUTER SYSTEMS (ICECCS 2008), p. 193–202. IEEE, 2008.

75 STEINMACHER, I.; CHAVES, A. P. ; GEROSA, M. A.. **Awareness support in distributed software development: A systematic review and mapping of the literature**. Computer Supported Cooperative Work (CSCW), 22(2-3):113–158, 2013.

76 KITCHENHAM, B.; BRERETON, O. P.; BUDGEN, D.; TURNER, M.; BAILEY, J. ; LINKMAN, S.. **Systematic literature reviews in software engineering–a systematic literature review**. Information and software technology, 51(1):7–15, 2009.

77 JAMI, S. I.; SHAIKH, Z. A.. **An autonomous provenance tracking system for collaborative environment**. 2011.

78 PIGNOTTI, E.; POLHILL, G. ; EDWARDS, P.. **Using provenance to analyse agent-based simulations**. In: PROCEEDINGS OF THE JOINT EDBT/ICDT 2013 WORKSHOPS, EDBT '13, p. 319–322, New York, NY, USA, 2013. ACM.

79 PIGNOTTI, E.; POLHILL, G. ; EDWARDS, P.. **Prov-o provenance traces from agent-based social simulation**. In: PROCEEDINGS OF THE JOINT EDBT/ICDT 2013 WORKSHOPS, EDBT '13, p. 333–334, New York, NY, USA, 2013. ACM.

80 DAVIS, D. B.. **Data Provenance for Multi-Agent Models in a Distributed Memory**. PhD thesis, 2017.

81 DAVIS, D. B.; FEATHERSTON, J.; FUKUDA, M. ; ASUNCION, H. U.. **Data provenance for multi-agent models**. In: 2017 IEEE 13TH INTERNATIONAL CONFERENCE ON E-SCIENCE (E-SCIENCE), p. 39–48. IEEE, 2017.

82 KICZALES, G.; LAMPING, J.; MENDHEKAR, A.; MAEDA, C.; LOPES, C.; LOINGTIER, J.-M. ; IRWIN, J.. **Aspect-oriented programming**. In: EUROPEAN CONFERENCE ON OBJECT-ORIENTED PROGRAMMING, p. 220–242. Springer, 1997.

83 WAMPLER, D.. **Aspect-oriented design principles: Lessons from object-oriented design**. In: SIXTH INTERNATIONAL CONFERENCE ON ASPECT-ORIENTED SOFTWARE DEVELOPMENT, 2007.

84 MILANÉS, A.; RODRIGUEZ, N. ; SCHULZE, B.. **State of the art in heterogeneous strong migration of computations**. Concurrency and Computation: Practice and Experience, 20(13):1485–1508, 2008.

85 KICZALES, G.; HILSDALE, E.; HUGUNIN, J.; KERSTEN, M.; PALM, J. ; GRISWOLD, W. G.. **An overview of aspectj**. In: EUROPEAN CONFERENCE ON OBJECT-ORIENTED PROGRAMMING, p. 327–354. Springer, 2001.

86 MARINS, A.; CASANOVA, M.; FURTADO, A. ; BREITMAN, K.. **Modeling provenance for semantic desktop applications**. SEMISH-XXXIV Seminário Integrado de Software e Hardware, Rio de Janeiro, RJ, SBC, 2007.

87 BUNGE, M.. **Treatise on basic philosophy: Ontology II: A world of systems**, volumen 4. Springer Science & Business Media, 2012.

88 ALVARES, L. O.; SICHMAN, J. S.. **Introduçao aos sistemas multiagentes**. In: XVII CONGRESSO DA SBC-ANAIS JAI'97, 1997.

89 BRATMAN, M.. **Intention, plans, and practical reason**, volumen 10. Harvard University Press Cambridge, MA, 1987.

90 BELHAJJAME, K.; B'FAR, R.; CHENEY, J.; COPPENS, S.; CRESSWELL, S.; GIL, Y.; GROTH, P.; KLYNE, G.; LEBO, T.; MCCUSKER, J. ; OTHERS. **Prov-dm: The prov data model**. W3C Recommendation, 2013.

91 LEBO, T.; SAHOO, S.; MCGUINNESS, D.; BELHAJJAME, K.; CHENEY, J.; CORSAR, D.; GARIJO, D.; SOILAND-REYES, S.; ZEDNIK, S. ; ZHAO, J.. **Prov-o: The prov ontology**. W3C recommendation, 2013.

92 DE CARVALHO, R. M.. **Programação orientada a aspectos**. <https://www.ricardomcarvalho.pt/blog/programacao-orientada-a-aspectos/>, 2013. [Online; Acessado em: 12-Novembro-2019].

93 WAINER, J.; OTHERS. **Métodos de pesquisa quantitativa e qualitativa para a ciência da computação**. Atualização em informática, 1:221–262, 2007.

94 ARAÚJO, M.; BARROS, M.; TRAVASSOS, G. ; MURTA, L.. **Métodos estatísticos aplicados em engenharia de software experimental**. XXI SBBD-XX SBES, 2006.

95 BATTISTI, I. D. E.; BATTISTI, G.. **Métodos estatísticos**. 2008.

96 DYBÅ, T.; KAMPENES, V. B. ; SJØBERG, D. I.. **A systematic review of statistical power in software engineering experiments**. Information and Software Technology, 48(8):745–755, 2006.

97 LIM, C.; LU, S.; CHEBOTKO, A. ; FOTOUHI, F.. **Prospective and retrospective provenance collection in scientific workflow environments**. In: 2010 IEEE INTERNATIONAL CONFERENCE ON SERVICES COMPUTING, p. 449–456. IEEE, 2010.

98 TRAVASSOS, G. H.; GUROV, D. ; AMARAL, E.. **Introdução à engenharia de software experimental**. UFRJ, 2002.

99 SIRQUEIRA, T. F. M.; DE LUCENA, C. J. P.. **Capturando e analisando proveniência de dados em sistemas multiagentes**. Monografias em Ciência da Computação–PUC-Rio, 2017.

100 SIRQUEIRA, T. F. M.; DE LUCENA, C. J. P.. **Proveniência de dados no bdi4jade: Um exemplo prático.** Monografias em Ciência da Computação–PUC-Rio, 2017.

101 SIRQUEIRA, T. F. M.; VIANA, M. L.; NASCIMENTO, N. ; DE LUCENA, C. J. P.. **Um framework para proveniência de dados**. Monografias em Ciência da Computação–PUC-Rio, 2017.

102 SIRQUEIRA, T. F. M.; VIANA, M. L. ; DE LUCENA, C. J. P.. **Proveniência de dados em sistemas multiagentes**. Monografias em Ciência da Computação–PUC-Rio, p. 11, 2017.

103 SIRQUEIRA, T. F. M.; VIANA, M. L. ; DE LUCENA, C. J. P.. **Capturando proveniência de dados em sistemas multiagentes**. Monografias em Ciência da Computação–PUC-Rio, p. 10, 2017.

A
Termo de Consentimento

TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO

1- Identificação do responsável pela execução da pesquisa:

Pesquisa: Sistemas Autônomos Explicáveis por meio de Proveniência de Dados.
Pesquisador: Tassio Ferenzini Martins Sirqueira
E-mail: tassio@tassio.eti.br

2- Informações ao participante:

1. Você está sendo convidado a participar de uma pesquisa que tem como objetivo verificar a eficácia do uso de ferramentas computacionais no âmbito de sistemas autônomos.
2. Antes de aceitar participar da pesquisa, leia atentamente as explicações abaixo, que informam sobre o procedimento:
   - Os participantes serão divididos em três grupos, sendo que o primeiro grupo avaliará o sistema de *logs,* o segundo grupo avaliará o sistema desenvolvido e o último avaliará ambos os sistemas;
   - Serão propostas, ao todo, três atividades, todas sob orientação do pesquisador.
   - As atividades serão feitas individualmente.
3. Você poderá recusar a participação na pesquisa e poderá abandonar o procedimento em qualquer momento, sem nenhuma penalização ou prejuízo. Durante os procedimentos, você poderá se recusar a responder qualquer pergunta que, por ventura, causar-lhe algum constrangimento.
4. A sua participação como voluntário não auferirá nenhum privilégio, seja ele de caráter financeiro ou de qualquer outra natureza, podendo se retirar da pesquisa a qualquer momento, sem seu prejuízo.
5. A sua participação não envolverá riscos, uma vez que não haverá exposição de sua imagem, nem a divulgação de sua identidade.
6. Serão garantidos, como afirmado anteriormente, o sigilo e privacidade dos envolvidos na pesquisa, sendo reservado ao participante o direito de omissão de sua identificação ou de dados que possam comprometê-lo.
7. Na apresentação dos resultados, não serão citados os nomes dos participantes.

Confirmo ter conhecimento do conteúdo deste termo. A minha assinatura abaixo indica que concordo em participar desta pesquisa e por isso dou meu consentimento.

_____, _____de _____ de 20_____.

Participante: _____

Assinatura: _____

B
Formulário de Caracterização

FORMULÁRIO DE CARACTERIZAÇÃO DO PARTICIPANTE DO ESTUDO

Número (Não preencher): _____

1) **Idade: _____ .**
2) **Gênero:** [ ] Masculino; [ ] Feminino; [ ]; [ ] Outros.
3) **Formação:**

   [ ] Pós-Doutorado; [ ] Doutorado; [ ] Mestrado; [ ] Especialização; [ ] Graduação; [ ] Técnico; [ ] Outra: _____ .

4) **Experiência:**

   [ ] Academia; [ ] Indústria; [ ] Academia e Industria.

   Tempo na academia (em meses): _____ .

   Tempo da indústria (em meses): _____ .

5) **Experiência em Desenvolvimento de Software**
   a) Nenhuma;
   b) Conhecimento teórico (estudei em aula ou em livro ou apliquei em projeto em sala de aula);
   c) Participei de um a dois projetos na indústria, mas como estagiário;
   d) Participei de um a dois projetos na indústria, como analista;
   e) Participei em mais de 03 projetos na indústria como analista ou cargo superior.
6) **Conhece sobre proveniência de dados?**

   [ ] Não; [ ] Sim.

7) **Conhece sobre sistemas multiagentes?**

   [ ] Não; [ ] Sim.

8) **Faz uso de algum sistema de logs em seus softwares?**

   [ ] Não; [ ] Sim.

C
Formulário de Avaliação Objetiva

FORMULÁRIO DE AVALIAÇÃO OBJETIVA:

Número (Não preencher): _____

Respondido observando:

[ ] *Logs*; [ ] Dados de Proveniência; [ ] *Logs* + Dados de Proveniência

Questão 1 - Hora de Início: ___:___ Hora de Término: ___:___

Quantos agentes foram iniciados com o sistema? _____

Questão 2 - Hora de Início: ___:___ Hora de Término: ___:___

Quando foi inicializado o sistema? _____

Questão 3 - Hora de Início: ___:___ Hora de Término: ___:___

Quando foi criado o agente chamado “HelloWorldAgent”? _____

Questão 4 - Hora de Início: ___:___ Hora de Término: ___:___

A agente denominada “Alice” é de qual tipo? _____

Questão 5 - Hora de Início: ___:___ Hora de Término: ___:___

Quantos agentes trocaram mensagens entre si? _____

Questão 6 - Hora de Início: ___:___ Hora de Término: ___:___

Em que horário o sistema foi finalizado? _____

Questão 7 - Hora de Início: ___:___ Hora de Término: ___:___

Em que horário foi criado o último container? _____

Questão 8 - Hora de Início: ___:___ Hora de Término: ___:___

Em que momento o agente do tipo “MultipleCapabilityAgent” foi iniciado? _____

Questão 9 - Hora de Início: ___:___ Hora de Término: ___:___

Quando ocorreu a primeira troca de mensagem (ping) no sistema? _____

Questão 10 - Hora de Início: ___:___ Hora de Término: ___:___

Quando ocorreu a última troca de mensagem (pong) no sistema? _____

Questão 11 - Hora de Início: ___:___ Hora de Término: ___:___

Em que hora aconteceu o evento “Socket closed”? _____

Questão 12 - Hora de Início: ___:___ Hora de Término: ___:___

Em que hora aconteceu o evento “sleep interrupted”? _____

Questão 13 - Hora de Início: ___:___ Hora de Término: ___:___

Quantas vezes a mensagem “ping” foi enviada? _____

Questão 14 - Hora de Início: ___:___ Hora de Término: ___:___

A que se refere o ID 5dbf59f95683e01b63e6b666? _____

Questão 15 - Hora de Início: ___:___ Hora de Término: ___:___

Ocorreu algum *Exception* no sistema? _____

Questão 16 - Hora de Início: ___:___ Hora de Término: ___:___

Ocorreu algum *Throwable* no sistema? _____

Questão 17 - Hora de Início: ___:___ Hora de Término: ___:___

Qual o endereço IP do servidor, responsável pela troca de mensagens Ping-Pong? _____

Questão 18 - Hora de Início: ___:___ Hora de Término: ___:___

Quais agentes estavam envolvidos na atividade “Send Message” do HelloWorldAgent? _____

Questão 19 - Hora de Início: ___:___ Hora de Término: ___:___

Qual o intervalo de tempo entre o início do sistema e a criação do último container? _____

Questão 20 - Hora de Início: ___:___ Hora de Término: ___:___

Qual o erro mais comum apresentado no sistema? Caso exista. _____

D
Formulário de Avaliação Qualitativa

FORMULÁRIO DE AVALIAÇÃO QUALITATIVA:

Número (Não preencher): _____

Respondido observando:

[ ] *Logs*; [ ] Dados de Proveniência; [ ] *Logs* + Dados de Proveniência

Questão 1) As informações relativas ao tempo de início, término de cada ação, foram facilmente identificadas a partir dos dados disponibilizados.

a. Discordo totalmente
b. Discordo parcialmente
c. Indiferente
d. Concordo parcialmente
e. Concordo totalmente

Questão 2) As informações contidas no detalhamento de cada ação, auxiliam no entendimento do que ocorreu durante a execução do sistema.

a. Discordo totalmente
b. Discordo parcialmente
c. Indiferente
d. Concordo parcialmente
e. Concordo totalmente

Questão 3) As informações contidas no detalhamento de uma "ActedOnBehalfOf", auxiliam na identificação dos agentes envolvidos em uma atividade.

a. Discordo totalmente
b. Discordo parcialmente
c. Indiferente
d. Concordo parcialmente
e. Concordo totalmente

Questão 4) Utilizar um modelo de dados de proveniência, auxilia na análise dos dados.

a. Discordo totalmente

b. Discordo parcialmente

c. Indiferente

d. Concordo parcialmente

e. Concordo totalmente

Questão 5) Possuir um *framework* para armazenar as informações das atividades de um sistema autônomo seguindo um modelo padrão, auxilia na análise dos dados.

a. Discordo totalmente

b. Discordo parcialmente

c. Indiferente

d. Concordo parcialmente

e. Concordo totalmente

Questão 6) Você considera importante compreender as ações realizadas por sistemas autônomos? Justifique

_____
_____
_____
_____
_____

Questão 7) Quais outras informações você julga importante e que não observou nos dados coletados?

_____
_____
_____
_____
_____

Questão 8) Existir um *framework* que auxilia na capturar os dados, seguindo um modelo de proveniência, apresenta quais vantagens ao desenvolvedor de sistemas autônomos ao seu ver?

_____
_____
_____

_____
_____

Questão 9) Quais os pontos positivos você classifica para o Prov-BDI4JADE?

_____
_____
_____
_____
_____

Questão 10) Quais os pontos negativos você classifica para o Prov-BDI4JADE?

_____
_____
_____
_____
_____

E
Histograma do Tempo de Reposta para cada Questão

E.1(a): Tempo para responder a Questão 1

E.1(b): Tempo para responder a Questão 2

E.1(c): Tempo para responder a Questão 3

E.1(d): Tempo para responder a Questão 4

E.1(e): Tempo para responder a Questão 5

E.1(f): Tempo para responder a Questão 6

E.1(g): Tempo para responder a Questão 7

E.1(h): Tempo para responder a Questão 8

E.1(i): Tempo para responder a Questão 9

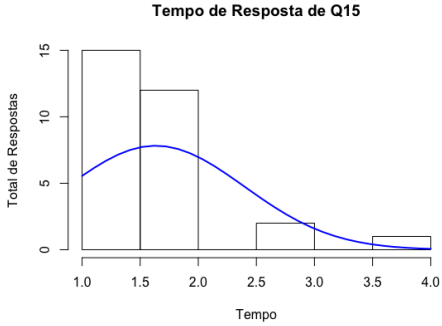
E.1(j): Tempo para responder a Questão 10

E.1(k): Tempo para responder a Questão 11

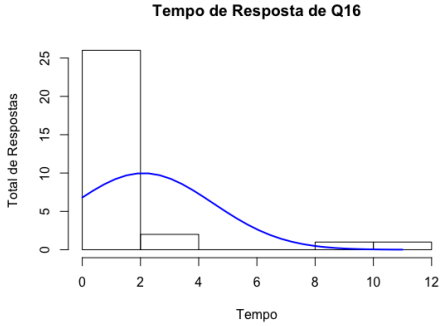
E.1(l): Tempo para responder a Questão 12

E.1(m): Tempo para responder a Questão 13

E.1(n): Tempo para responder a Questão 14



E.1(o): Tempo para responder a Questão 15



E.1(p): Tempo para responder a Questão 16

E.1(q): Tempo para responder a Questão 17

E.1(r): Tempo para responder a Questão 18

E.1(s): Tempo para responder a Questão 19

E.1(t): Tempo para responder a Questão 20

Figura E.1: Histograma do Tempo de Reposta para cada Questão.