UNIVERSIDADE FEDERAL DO PAMPA

Lukas Felipe Gaedicke

Extensão de um Metamodelo para a Especificação de Requisitos em um Contexto de Agentes de Tempo Real

Alegrete
2019

Lukas Felipe Gaedicke

Extensão de um Metamodelo para a Especificação de Requisitos em um Contexto de Agentes de Tempo Real

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação Engenharia de Software da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Engenharia de Software.

Orientador: Prof. Dr. Gilleanes Thorwald Araujo Guedes

Alegrete
2019

Lukas Felipe Gaedicke

Extensão de um Metamodelo para a Especificação de Requisitos em um Contexto de Agentes de Tempo Real

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação Engenharia de Software da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Engenharia de Software.

Trabalho de Conclusão de Curso defendido e aprovado em 28 de NOVEMBRO de 2019
Banca examinadora:

Prof. Dr. Gilleanes Thorwald Araujo Guedes
Gilleanes Thorwald Araujo Guedes
UNIPAMPA

Prof. Dr. João Pablo Silva da Silva
UNIPAMPA

Prof. Dr. Maicon Bernardino da Silveira
UNIPAMPA

Humildemente dedicado a:
Marcos, a quem chamo de “pai”,
Silvana a quem chamo de “mãe”,
Meus irmãos Bruno e Jeny,
Aos meus avós a quem chamo de “Opa” e “Oma”.

AGRADECIMENTOS

Agradeço primeiramente aos meus pais e avôs (Opa e Oma) por todo o auxilio durante a graduação, sem vocês, o final dela não teria sido possível.

Para Gillanes, um agradecimento por me orientar ao longo desses quase 4 anos de iniciação cientifica e pelas mais diversas conversas entre as pausas das reuniões de orientação.

Para Luiz Paulo, meu irmão de coração, um agradecimento eterno por me apresentar Alegrete, e para Anne, agradecimento especial a por ter me auxiliado e me matricular no curso.

Para Mel, agradeço por sempre me apoiar e revisar todas minhas empreitadas.

Aos "colonos" Paulo, Marquin, Gili (pelas fubadas), Jojo, Alex e o menino Wolly agradeço pela amizade, as longas conversas e festas. Vocês sem dúvida fizeram minha passagem por Alegrete ser repleta de momentos memoráveis.

Aos amigos Jake, Jean, André, Dionatã e Jonner agradeço a vocês pela parceria e suporte ao longo desses anos de curso.

Do you know that story of the Russian cosmonaut?
So, he goes up in this big spaceship
And he's got this portal window
And he's looking out of it
And he sees the curvature of the Earth for the first time
And all of a sudden, this strange ticking
Begins coming out of the dashboard (Okay, yeah)
But he can't find it, he can't stop it, it keeps going
A few hours into this, it begins to feel like torture
What's he gonna do? He's up in space!
So the cosmonaut decides
The only way to save his sanity
Is to fall in love with this sound
(Spoken: Brit Marling and William Mapother on film Another Earth)

RESUMO

A demanda por aplicações de software, denominadas sistemas multiagentes, que contenham entidades autônomas com capacidade de operar e tomar decisões sem intervenções externas está cada vez maior. No entanto, existem desafios no desenvolvimento desse tipo de sistema, tais como a especificação correta de requisitos. Embora essa seja uma questão crucial no desenvolvimento de qualquer software, sistemas multiagentes possuem requisitos particulares, posto que os agentes que os compõem devem ser autônomos e, quando seguindo o modelo BDI, devem possuir crenças, objetivos e planos, bem como serem capazes de perceber o ambiente e agir sobre ele. Quando um agente (em geral interpretando um papel) acredita que um objetivo pode ser atingido, ele passa a agir para realizar esse objetivo, em geral por meio da execução de um plano. Porém, apenas a intenção de atingir um objetivo não garante que o objetivo será concluído, muito menos que ele será atingido dentro do tempo esperado. Nesse sentido, pesquisadores começaram a utilizar agentes de tempo real para solucionar problemas afetados por restrições temporais. Com o intuito de abordar o problema de especificação desse tipo de sistema, alguns estudos estenderam as metaclasses utilizadas para produzir diagramas de casos de uso da UML de forma a representar requisitos particulares de sistemas multiagentes. Contudo, através de um mapeamento sistemático, identificamos que atualmente a maioria das propostas ainda estão projetando seus sistemas sem levar em conta as restrições temporais associados aos objetivos e outros comportamentos associados a papeis de agente. Assim, neste trabalho realizamos uma extensão de um metamodelo de agência para a representação de restrições temporais em requisitos específicos para papeis de agente em tempo real. Além disso, foi realizada a instanciação do metamodelo, em conjunto de regras OCL, que restringem as associações apenas entre agentes e seus comportamentos em tempo real. Como forma de avaliação, um experimento foi realizado, por meio de um grupo focal, com alunos de pós-graduação.

Palavras-chave: Sistemas Multiagentes. Engenharia de Requisitos. Diagrama de Casos de uso. Agentes de tempo real.

ABSTRACT

The demand for software applications, called multi-agent systems containing autonomous entities with the capability of operating and taking decisions without external interventions is growing. However, there are challenges in the development of this kind of system, such as the correct requirements specification. Though this is a crucial question in the development of any software, multi-agent systems own particular requirements, since the agents that compose it must be autonomous and, when following the BDI model, they must own beliefs, goals, and plans, as well as be able to perceive the environment and act upon it. When a agent (in general interpreting a role) believe that a goal can be achieved, he pass to act to accomplish this goal, generally by means of the execution of a plan. However, only the intention to achieve a goal does not guarantee that the goal will be concluded, much less that it will be achieved in the expected time. This way, researchers are beginning to use real-time agents to solve troubles affected by temporal constraints. In order to address the requirements specification of this kind of system, some studies had extended the metaclasses used to produce UML use case diagrams in order to represent multi-agent systems particular requirements. However, by means of a systematic mapping, we identified that, presently, mostly of the proposals are still designing their systems without considering temporal restrictions associated to the goals and other behaviors of agent roles. Thus, in this paper we performed an extension of an agency metamodel to represent temporary constraints on requirements applicable to real-time agent roles. In addition, a metamodel was installed in the OCL rule set, which restricted associations only between agents and their real-time results. As an evaluation, an experiment was performed, by means of a focus group, with master degree students.

Key-words: Multi-Agent Systems. Requirements Engineering. Use-Case Diagrams. Real-Time Agents.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

ART Agente de Tempo Real

BDI Belief–Desire–Intention

DCU Diagrama de Casos de Uso

ER Engenharia de Requisitos

MSL Mapeamento Sistemático da Literatura

OCL Object Constraint Language

SMA Sistema Multiagente

SMA-RT Sistema multiagente de tempo real

UML Unified Modeling Language – Linguagem de Modelagem Unificada

SUMÁRIO

1 INTRODUÇÃO

Ao longo dos anos, pesquisadores na área de Engenharia de Software criaram métodos de desenvolvimento e linguagens de modelagem de forma a produzir software com mais qualidade, melhor estruturado, mais confiável e fácil de manter e evoluir. Já na área de Inteligência Artificial, o uso de agentes como auxiliares em software aplicado a diversos domínios tem se espalhado (DORRI; KANHERE; JURDAK, 2018), (JULIAN; BOTTI, 2019), (VIRUEL, 2011) e o interesse da indústria por sistemas multiagentes tem crescido (SLHOUB; CARVALHO; BOND, 2017). Esse tipo de software tem demonstrado ser uma boa alternativa para o desenvolvimento de sistemas complexos (BOES; MIGEON, 2017), (DORRI; KANHERE; JURDAK, 2018), (NAKAGAWA et al., 2010), levando a um aumento de seu desenvolvimento (JULIAN; BOTTI, 2019).

Todavia, o desenvolvimento de sistemas multiagentes apresentou novos desafios para a Engenharia de Software (DORRI; KANHERE; JURDAK, 2018). Isso levou ao surgimento da *Agent-Oriented Software Engineering* (AOSE), uma área que mistura características da Engenharia de Software e da Inteligência Artificial. Essa área visa adaptar as práticas de Engenharia de Software especificamente para o desenvolvimento desse tipo de sistema (GUEDES; VICARI, 2011).

A área de AOSE incentivou pesquisadores a proporem a criação de novas processos de desenvolvimento e linguagens de modelagem para esse tipo de sistema. Dentro desse contexto, uma das diversas subáreas da engenharia de software é a engenharia de requisitos, que visa extrair, analisar, especificar e validar os requisitos dos sistemas, de modo a permitir a compreensão correta do que realmente um sistema deve fazer (BOURQUE; FAIRLEY et al., 2014). Dessa forma, esforços foram feitos para adaptar esta subárea para especificar requisitos específicos para sistemas multiagentes.

Sistemas multiagentes são compostos por vários agentes que interagem entre si (WOOLDRIDGE, 2009). De acordo com Vicari e Gluz (2007) e Wooldridge e Jennings (1995), um agente é um processo autônomo, flexível, proativo que pode atuar em seu ambiente sem ser comandado por atores externos e, eventualmente, cooperar com outros agentes para atingir um ou mais objetivos. Os agentes ainda podem vir a interpretar papéis, os quais especificam as abstrações relacionadas com as capacidades e comportamentos a serem exigidos por um ou mais agentes que irão desempenhar esse papel (TRENCANSKY; CERVENKA, 2005).

Agentes inteligentes podem ser classificados de diversas maneiras, como por exemplo, por suas arquiteturas e suas restrições de tempo. As arquiteturas, de acordo com Russell e Norvig (2016) definem a estrutura interna do agente, estabelecendo como o agente irá perceber e atuar sobre o ambiente em que está localizado. Já as restrições de tempo definem, segundo Qasim (2017), que um agente de tempo real deve tentar atingir seus objetivos dentro das restrições temporais impostas, caso contrário o objetivo não terá sido atingido realmente.

Contudo, para Dragoni, Sernani e Calvaresi (2018), um agente definido apenas como inteligente é somente uma ideia esboçada em um quadro branco. Para os autores, um agente só é inteligente se for um agente de tempo real, percebendo o tempo de alguma maneira. Assim, de acordo com estes autores, ser inteligente não implica apenas raciocinar sobre o tempo, mas também raciocinar no tempo.

Dessa forma, agentes de tempo real devem executar seus comportamentos dentro das restrições de tempo especificadas, caso contrário seus objetivos não serão atingidos e os agentes irão perder sua usabilidade. Nesse sentindo, sistemas complexos que fazem uso de agentes com características temporais estão vulneráveis a restrições de tempo. Assim, reconhecendo a importância de fornecer mecanismos para representar as noções de agentes temporais em processos de desenvolvimento de sistemas multiagentes, este trabalho estendeu o metamodelo de agência de Guedes e Vicari (2011), criando novas metaclasses, além de regras OCL, para representar e restringir a representação de funcionalidades de agentes associadas a restrições temporais.

1.1 Objetivos

Com base nas limitações encontradas na literatura em relação a especificação de requisitos para sistemas multiagentes de tempo real, esse trabalho teve como objetivo desenvolver mecanismos para permitir a identificação de requisitos específicos para sistemas multiagentes de tempo real. Nesse sentindo, os principais objetivos desse estudo são:

- Identificar as principais noções de agência de tempo real;
- Desenvolver um mapeamento sistemático para identificar como os requisitos inerentes aos sistemas multiagentes são representados;
- Estender o metamodelo de Guedes e Vicari (2011) para representar funcionalidades em tempo real;
- Validar a proposta.

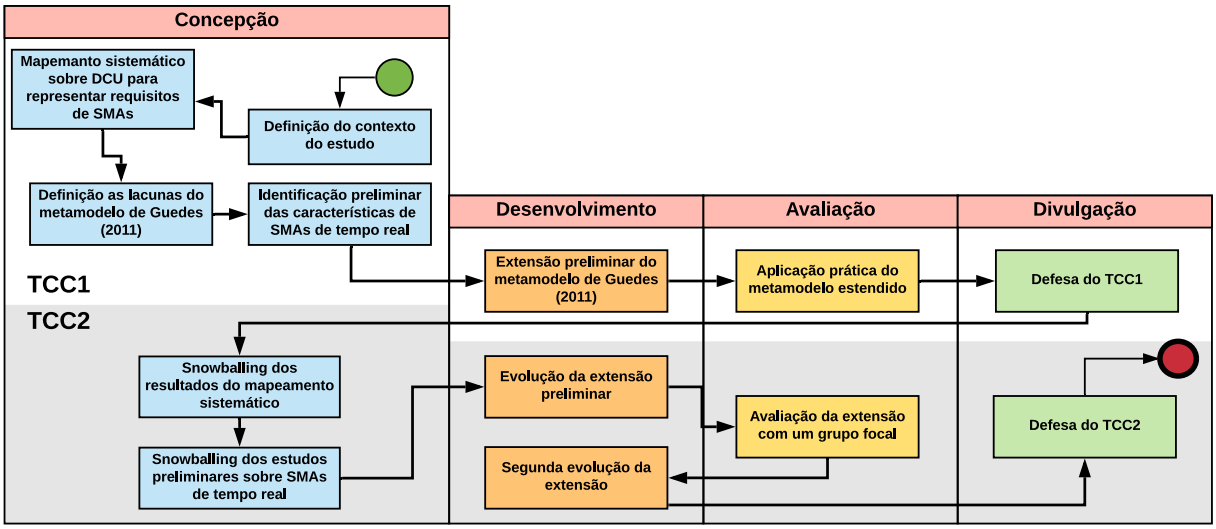
1.2 Metodologia

Patten e Newhart (2017) descrevem métodos de pesquisa como blocos que colaboram na construção do empreendimento científico. As metodologias possuem um conjunto dos melhores métodos que colaboram para descrever a forma de “como” as coisas devem ser realizadas para o conhecimento sistemático ser construído.

Dessa forma, para um estudo reduzir seus possíveis vieses e possuir resultados com bons níveis de qualidade e confiáveis, é indispensável utilizar uma metodologia de pesquisa bem definida. Dessa forma, planejamos conduzir esse estudo em 4 fases (Concepção, Desenvolvimento, Avaliação, Divulgação). Em cada um dessas fases foram definidos marcos

para representar pontos de chegada que deveriam ser atingidos para a conclusão deste trabalho. Essas etapas podem ser vistas na Figura 1.

Figura 1: Ilustração da metodologia utilizada durante o estudo.



Fonte: O autor.

1.3 Organização

O restante desse estudo está organizado da seguinte forma:

- Capítulo 2 apresenta as noções sobre agência em que conceitos relacionados as características dos agentes, arquiteturas internas dos agentes, seus ambientes e assim como suas organizações são descritos;
- Capítulo 3 descreve a estratégia de pesquisa utilizada em nossa revisão bibliográfica, bem como seus resultados, que serviram de base para o presente trabalho;
- Capítulo 4 apresenta nossa extensão para representar sistemas multiagentes de tempo real;
- Capítulo 5 descreve a validação da nossa extensão por meio de um grupo focal;
- Capítulo 6 descreve as considerações finais deste estudo.

2 FUNDAMENTAÇÃO TEÓRICA E TECNOLÓGICA

Nesta seção são apresentados e discutidos os fundamentos e teorias que são encontrados e utilizados nos capítulos e seções seguintes deste estudo e que fazem parte do domínio de sistemas multiagentes de tempo real, servindo assim como base para a compreensão deste trabalho.

2.1 Agentes

Russell e Norvig (2016) consideram agente qualquer coisa que possa perceber um ambiente através de sensores e atuar sobre esse ambiente através de atuadores. Todavia, outros autores apresentam definições mais detalhadas.

Wooldridge e Jennings (1995) declaram que um agente pode ser definido como um processo computacional baseado em software que contenha as seguintes propriedades:

- **Autonomia**: Define que os agentes contêm a capacidade de operar sobre um ou mais ambientes sem qualquer intervenção de humanos, além de possuírem algum tipo de controle sobre suas ações e estados internos;
- **Capacidade social:** Descreve que os agentes possuem capacidades de realizar interações com humanos ou outros agentes situados no ambiente;
- **Reatividade**: Define a percepção dos agentes sobre seu ambiente e sua capacidade em responder de forma oportuna às mudanças que ocorrem nele;
- **Proatividade:** Descreve que os agentes podem ser capazes de exibir um comportamento direcionado por objetivos.

Ainda para Wooldridge e Jennings (1995) *apud* (SHOHAM, 1993), um agente pode também conter definições mais complexas, em que, além do agente ser um sistema computacional e conter as propriedades listadas acima, um agente pode possuir características que também podem ser aplicados a seres humanos, como conhecimento, crenças, intenções e obrigações. Outros autores definem outras características que podem compor um agente, sendo elas:

- **Emocionais:** De acordo com Bates et al. (1994), os objetivos e as avaliações dos eventos realizados por um agente podem produzir um estado emocional. E por meio desse estado, é possível moldar a maneira como o agente irá reagir a esses eventos;
- **Confiabilidade:** Galliers (1988) e Chin et al. (2014) descrevem a característica de que um agente não irá comunicar intencionalmente informações falsas;
- **Benevolência:** Rosenschein e Genesereth (1988) e Chin et al. (2014) trabalham com a suposição de que os agentes não possuem objetivos conflitantes, e que esses

agentes troquem dados e hipóteses a fim de cooperar mutuamente na execução de tarefas comuns evitando a interferência durante o uso de recursos comuns, bem como realizem as tarefas que lhes são solicitadas;

- **Racionalidade:** Galliers (1988) e Chin et al. (2014) descrevem a racionalidade de um agente, como o compromisso do mesmo em atingir seus objetivos, além de estabelecer que o agente não irá agir de maneira à impedir que seus próprios objetivos sejam alcançados;

- **Capacidade de aprender:** Chin et al. (2014) descreve a capacidade de um agente adaptar-se a um ambiente e aos desejos de seus usuários;

- **Mobilidade:** Chin et al. (2014) descreve a capacidade de um agente em se locomover entre diferentes dispositivos em uma rede de computadores.

Já para Wooldridge (2009), um agente é uma entidade de software autônoma que está situada em um ambiente, no qual pode monitorar e responder a mudanças de forma proativa ou reativa por si mesma ou por meio de comunicação com outros agentes para alcançar persistentemente determinado objetivo ou tarefa em nome do usuário ou de outros agentes.

A noção de agência considerada para esse estudo, é a declarada por Vicari e Gluz (2007), em que os objetivos de um agente podem ser definidos através de suas crenças e desejos, e o seu comportamento são fortemente influenciados por suas intenções. Além disso, os agentes podem perceber eventos que ocorrem no ambiente em que está localizado, e realizar ações sobre o mesmo. Os agentes também podem ser classificados em reativos ou cognitivos, os reativos apenas reagem a eventos, enquanto os cognitivos possuem crenças, desejos e intenções a respeito do ambiente em que se encontram.

2.1.1 Agentes *Belief–Desire–Intention*

Em linhas gerais um agente segue o modelo Belief–Desire–Intention (BDI) quando seu comportamento é baseado em estados mentais como crenças, desejos e intenções, ou em outras palavras, nessa arquitetura um agente pode ser totalmente especificado pela definição de suas crenças e desejos e o seu comportamento é fortemente influenciado por suas intenções (VICARI; GLUZ, 2007).

Em resumo, em uma arquitetura BDI:

- As crenças são o conjunto de informações que um agente tem sobre o mundo, ou ainda, o que ele acredita ser verdade sobre o ambiente em que ele se encontra;

- Os desejos são o que o agente almeja alcançar, ou seja, seus objetivos;

- As intenções representam compromissos do agente para com os seus desejos e crenças, ou seja, uma intenção determina que um agente acredita que um desejo pode ser atingido de acordo com suas crenças e assim ele passará a agir para realizá-lo.

2.1.2 Agentes em Tempo Real

Em seu estudo Julian e Botti (2004) descreve que ART podem atuar em ambientes em que o tempo é um fator crítico. Esses ambientes requerem agentes que hajam de forma autônoma enquanto ainda trabalham em objetivos comuns. Esses agentes por sua vez exigem respostas em tempo real e devem eliminar a possibilidade de comunicação maciça entre os agentes.

Para DiPippo, Hodys e Thuraisingham (1999) Julian e Botti (2004) Król e Nowakowski (2013) Qasim (2017), brevemente, um agente de tempo real é um agente cujas ações podem ser limitadas por restrições de tempo. Já para (ASHAMALLA, 2017), em uma definição mais elaborada, os ART são agentes que levam em consideração o tempo para atingir seus objetivos. Além de concordarem com as definições anteriores, os autores Julian et al. (2002), Soler et al. (2002) Carrascosa et al. (2003) Carrascosa et al. (2005a) Carrascosa et al. (2004) descrevem que as restrições dos ART podem ser "hard" e/ou "soft", e que essa classificação, segundo Carrascosa et al. (2004), depende se os ART têm requisitos temporais criticos ou não. Também Carrascosa et al. (2005b), reforça que, se um agente de tempo real com restrições de tempo apresentar resultados insatisfatórios, podem haver consequências catastróficas, e perante isso o agente é classificado como "hard".

Outra definição é apresentada por DiPippo et al. (2001), que define que um agente de tempo real deve atingir seus objetivos dentro de restrições de tempo específicas, mas que ao tentar atingir seus objetivos pode possivelmente impactar a qualidade dos seus resultados. Além disso, Bajo et al. (2008) descreve que um agente de tempo real deve garantir o cumprimento de suas restrições de tempo e, ao mesmo tempo, deve tentar atingir suas metas ou objetivos. Nesse sentindo, Qasim, Kazmi e Fakhir (2015) afirma que para esse tipo de agente cumprir seus objetivos dentro dos prazos específicos é importante, já que o oposto disso faz com eles percam sua usabilidade.

Finalmente para Dragoni, Sernani e Calvaresi (2018), um agente de tempo real é um processo em execução cuja correção depende não apenas da totalidade de seu executável, mas também do momento em que a ação é "executada". Nesse sentido, um agente de tempo real é um processo em execução que atua em um ambiente e que precisa ser executado a tempo, possivelmente raciocinando explicitamente sobre o tempo. Fato esse que se faz necessário incorporar a ele um relógio (*timer* ou contador), além de que se o mesmo raciocina sobre o tempo, provavelmente suas ações/planos irão mudar dependendo da situação, em função do tempo atual/tempo restante para atingir seus objetivos. Podendo assim também alterar seus planos com base no requisitos de qualidade de seus

objetivos.

Como exemplo da aplicação de agência de tempo real, Dragoni, Sernani e Calvaresi (2018), descrevem um agente de tempo real implementado em um robô aspirador de pó. Esse robô é programado/treinado para limpar os cômodos de uma casa, no qual o mesmo deve sempre planejar a melhor sequência de cômodos a serem limpos, buscando a economia da sua energia, por que quando ela acaba o mesmo precisa interromper a limpeza e ir até a estação de carga. Nesse sentindo, esse agente precisa o tempo todo estar refletindo sobre o tempo, no qual ele precisa ter total controle sobre a disponibilidade sua carga e a mesma é suficiente para atingir seu objetivo.

Dragoni, Sernani e Calvaresi (2018) também apresentam um outro exemplo, consideravelmente mais complexo, proveniente do mercado de ações, no qual, um SMA-RT é responsável pela compra e venda de grandes quantidades de ações que devem ser realizadas a cada poucos milissegundos. Nesse sentindo, alguns dos ART devem monitorar o mercado, por exemplo, a cada 50ms. Já outros ART "agem" comprando ou vendendo ações a cada 350/700ms. Dessa forma, o sucesso dos objetivos desses ART são fortemente influenciados pela duração de suas decisões.

2.1.3 Papéis de Agente

Para Trencansky e Cervenka (2005), os papéis de agente são usados para definir um repertório comportamental dos agentes, em outras palavras, os papéis podem ser usados para modelar abstrações relacionadas com as capacidades, comportamentos, observações, relacionamentos e serviços a serem oferecidos ou exigidos de um agente em um contexto específico.

Já para Silva e Lucena (2004) um papel guia e também restringe o comportamento dos agentes que interpretam um papel. Ainda para Silva e Lucena (2004), um papel pode (i) descrever os objetivos, crenças e ações que um agente irá executar durante a interpretação de um papel; (ii) adicionar novas metas e crenças ao conjunto de metas e crenças associadas a um agente além de descrever os deveres, direitos e protocolos relacionados a um agente enquanto ele desempenha o papel.

É importante observar que um agente pode interpretar um ou mais papéis de agente, e quando o agente assumir interpretar um papel, o mesmo deve ser capaz de executar as ações e planos associados ao papel, além de possuir a intenção de atingir os objetivos propostos pelo papel (SILVA; LUCENA, 2004; SILVA, 2007).

2.2 Sistemas Multiagentes

Uma definição de SMA é descrita por Wooldridge (2009), que estabelece que um SMA é um sistema que contém um certo número de agentes que interagem entre si, geralmente trocando mensagens através de alguma infraestrutura de rede de computadores.

Wooldridge (2009) ainda destaca que geralmente os agentes em um SMA estarão agindo de formas diferentes para alcançarem seus objetivos e muitas vezes para completar seus objetivos os agentes terão que agir de maneira cooperativa entre si afim de realizarem ações e negociações conjuntas visando um objetivo comum.

Outra definição mais simples é apresentada por Dunin-Keplicz e Verbrugge (2011), em que SMA são sistemas computacionais nos quais uma coleção de agentes autônomos e fracamente acoplados interagem entre sí para resolver um determinado problema. O problema muitas vezes pode estar além das capacidades individuais dos agentes, assim os agentes acabam por explorar sua capacidade de se comunicar, cooperar, coordenar e negociar uns com os outros para atingir seus objetivos.

Já em um contexto de ART, Julian et al. (2002), Soler et al. (2002), Julian e Botti (2004), Bajo et al. (2008), Qasim (2017), definem que um SMA em tempo real é um sistema multiagente com pelo menos um agente em tempo real. Já para Ashamalla (2017) sistemas multiagentes em tempo real são SMA com restrições de tempo. Além disso, DiPippo et al. (2001) descreve que em um SMA em tempo real os agentes se comunicam, coordenam e negociam para atingir seus objetivos, dentro do prazo especificado e com restrições de qualidade. Já Bajo et al. (2008) acrescenta que, em um SMA em tempo real, as comunicações e protocolos entre agentes não podem ser muito complexos, já que podem impactar no desempenho dos agentes.

2.3 Especificação de Requisitos

A especificação de requisitos é uma das atividades cobertas pela engenharia de requisitos. De acordo com Berenbach et al. (2009), a engenharia de requisitos envolve todas as atividades do ciclo de vida do sistema dedicadas a identificar requisitos do usuário, a analisar requisitos para derivar requisitos adicionais, documentar requisitos por meio de uma especificação e validar os requisitos documentados.

De acordo com o Bourque, Fairley et al. (2014), na engenharia de software o termo especificação de requisitos de software, normalmente se refere à produção de um documento que pode ser revisado, avaliado e aprovado sistematicamente. O tamanho e a complexidade desse documento varia de escopo para escopo, em sistemas complexos são produzidos até três tipos diferentes de documentos: definição do sistema, requisitos do sistema e requisitos de software, já em sistemas mais simples apenas um terço desses documentos são necessários. Além disso, para a documentação dos requisitos, muitos autores recomendam o uso do diagrama de casos de uso UML (BERENBACH et al., 2009), (REGNELL, 1999), (SOMMERVILLE, 2011).

É importante destacar que em um projeto de software, um ou mais documentos de especificação são essenciais. De acordo com o Bourque, Fairley et al. (2014), a especificação estabelece a base para acordos entre clientes e contratados sobre o que o software deve ou não fazer. Além de que, a especificação de requisitos fornecer uma base realista para

estimar custos, riscos e cronogramas do software a ser produzido, assim permitindo uma avaliação rigorosa dos requisitos antes que o projeto possa começar e reduz o retrabalho posterior.

Vários autores, como Berenbach et al. (2009), Dick, Hull e Jackson (2017), e Regnell (1999), declaram que uma especificação correta de requisitos é essencial para o sucesso de um projeto de software, embora destacam que essa não é uma tarefa fácil. Assim, a engenharia de requisitos envolve todas as atividades do ciclo de vida do sistema dedicadas a identificar requisitos do usuário, analisar requisitos para derivar requisitos adicionais, documentar requisitos e validar os requisitos especificados (BERENBACH et al., 2009), (BOURQUE; FAIRLEY et al., 2014).

2.4 UML

De acordo com a OMG (2017), a UML é uma linguagem visual para especificar, construir e documentar os artefatos de sistemas. Ela é uma linguagem de modelagem de propósito geral que pode ser aplicada a todos os domínios de aplicação.

Segundo Booch, Jacobson e Rumbaugh (2016), a UML é uma linguagem adequada para a modelagem de sistemas, cuja abrangência poderá incluir desde sistemas de informação corporativos a serem distribuídos a aplicações baseadas na *Web* e até sistemas complexos embutidos de tempo real. Ela é uma linguagem muito expressiva, abrangendo todas as visões necessárias ao desenvolvimento e implantação desses sistemas.

A UML tornou-se, nos últimos anos, de acordo com Guedes e Vicari (2012), a linguagem padrão de modelagem adotada internacionalmente tanto nos meios acadêmicos como pela indústria de engenharia de software. Uma das vantagens da UML, é que a linguagem permite facilmente representar sistemas de forma padronizada facilitando assim a compreensão da pré-implementação dos sistemas propriamente ditos.

Outra vantagem da UML, é que ela permite modelar e documentar qualquer sistema, sendo uma linguagem de modelagem totalmente independente, não estando vinculada a nenhum processo ou metodologia de desenvolvimento específico.

2.4.1 Metamodelo, Perfil e Estereótipos

Um metamodelo define uma linguagem semântica capaz de fornecer instâncias para a criação de modelos (OMG, 2017). Guedes e Vicari (2012) descreve que o papel típico de um metamodelo é definir a semântica para a forma de modelar elementos dentro de um modelo sendo instanciado, dessa forma um modelo é uma instância de um metamodelo.

Guedes e Vicari (2012) descreve que um perfil é um tipo de pacote que estende um metamodelo de referência. Através de um perfil, é possível introduzir novas restrições por meio da adição ou modificação da semântica das metaclasses de um metamodelo já existente. No contexto da UML, os perfis são usados para permitir a adaptação da referida

linguagem afim de adequá-la a certos domínios, métodos ou tecnologias.

Já um estereótipo é definido por Guedes e Vicari (2012), como um tipo limitado de metaclasse que não pode ser utilizada unicamente, mas deve sempre ser usada em conjunção com uma das metaclasses que ele estende. Em um contexto de metamodelos e perfis, um estereótipo define como uma metaclasse pode ser estendida, atribuindo-lhe novas características e/ou restrições. Sendo assim, quando uma nova metaclasse é derivada a partir de uma metaclasse anterior e utiliza o estereótipo "stereotype", essa nova metaclasse permite modelar instâncias da metaclasse original contendo um estereótipo com o nome da nova metaclasse.

2.4.2 Diagrama de Casos de Uso

O Diagrama de Casos de Uso (DCU) pode ser utilizado para a especificação de requisitos. Em linhas gerais, o principal objetivo desse diagrama é possibilitar identificação dos tipos de usuários externos e a maneira como os mesmos irão interagir com o sistema. Além de ser possível, por meio dessa representação, identificar prematuramente possíveis falhas na especificação.

Para Guedes (2018), o DCU tem como objetivo apresentar uma visão externa geral das funcionalidades de um sistema a ser desenvolvido. Ainda para o autor, esse artefato por meio de uma linguagem simples possibilita que qualquer pessoa possa abstrair e identificar os requisitos que poderão compor um sistema.

Esse diagrama geralmente é criado nas etapas inicias do desenvolvimento de *software*, sendo essas as etapas de elicitação, análise e especificação, ainda sendo possível realizar a verificação e validação dos mesmos em busca de falhas na elicitação e na especificação do problema. Além disso, é importante destacar ainda que os DCU podem ser modificados durante todo ciclo de vida do desenvolvimento, não se limitando apenas as etapas inicias.

2.4.2.1 Atores

Os atores em um DCU para Guedes (2018), representam os diversos papéis desempenhados pelos usuários externos que de alguma forma irão utilizar o sistema. Os atores são representados nos DCU como "bonecos magros" contendo abaixo da sua representação uma descrição do papel assumido pelo usuário para realizar alguma função no sistema. A Figura 2 representa um ator no DCU da UML.

2.4.2.2 Casos de uso

Os casos de uso, ainda para Guedes (2018), são utilizados para representar as funcionalidades identificadas e definidas como necessárias para que os atores possam utilizar o sistema. Os casos de uso ainda podem ser classificados em casos de uso primários e

Figura 2: Representação de um ator no DCU da UML.

Fonte: (OMG, 2017).

secundários, no qual a primeira classificação é realizada para descrever funcionalidades referentes a algum processo importante focado nos requisitos do sistema a ser desenvolvido. Já os casos de uso secundários descrevem e representam os processos secundários, como operações básicas dos tipos de criação, consulta, atualização e exclusão de dados (CRUD). A representação gráfica dos casos de uso se dá por meio de elipses que contém em sí uma descrição da funcionalidade referente ao caso de uso. A Figura 3 representa um caso de uso na UML.

Figura 3: Representação de um caso de uso na UML.

Fonte: (OMG, 2017).

2.5 Metamodelo de Guedes

Como descrito anteriormente, o desenvolvimento de sistemas multiagentes possui características particulares. Sendo assim, as abordagens tradicionais para o desenvolvimento de software, muitas vezes não possuem todos os recursos necessários para fornecer suporte no desenvolvimento desse tipo sistema.

Nesse sentindo, um exemplo são os casos de uso apresentados pela UML, que apesar de serem amplamente utilizados para a engenharia de requisitos para softwares tradicionais, quando utilizado em um contexto de desenvolvimento de sistemas multiagentes, apresenta limitações em relação a representação de um agente no DCU. Um agente é um ator interno ao sistema, e de acordo com as diretrizes da OMG (2017), o sistema não pode ser representado nos DCU como um ator, que é apresentado como um ator externo ao sistema.

Assim considerando a importância da fase de análise de requisitos para um bom projeto de software e identificando uma lacuna relacionada aos mecanismos fornecidos pela UML para a modelagem de requisitos funcionais específicos para projetos de sistemas multiagentes Guedes e Vicari (2011) criou um metamodelo UML para solucionar essa lacuna.

No metamodelo estendido, os autores acrescentaram novas metaclasses e estereótipos ao metamodelo em que se baseia no DCU da UML, para identificar as funcionalidades particulares deste tipo de sistema. O metamodelo estendido por Guedes (2018), pode ser visto na Figura 4.

No metamodelo estendido por Guedes e Vicari (2011), o DCU da UML foi enriquecido com novas metaclasses, sendo elas:

- **AgentRole_Actor**: como descrito anteriormente, os agentes não são externos ao software, assim um ator no DCU não pode representar papéis de agente. Assim (GUEDES; VICARI, 2011), criou essa metaclasse para ser possivel representar um papel interpretado por um agente como algo interno ao software;

- **Reactive_AgentRole e Cognitive_AgentRole:** a partir da metaclasse criada anteriormente (AgentRole_Actor) (GUEDES; VICARI, 2011) derivou duas novas metaclasses para representar papéis interpretados por agentes cognitivos e reativos, sendo elas: *Reactive_AgentRole* e *Cognitive_AgentRole*. Além da extensão, aplicou-se a elas o estereótipo "stereotype", o que significa que estas metaclasses, para serem utilizadas, devem ser usadas como estereótipos sobre os *AgentRole_Actors*;

- **Papéis de agentes especialistas:** com base nas sugestões de Vicari e Gluz (2007), que descreve o uso dos agentes especialistas para o projeto de SMA, através das metaclasse PS_AgentRole, UAM_AgentRole e SMI_AgentRole extendidas da metaclass Cognitive_AgentRole, (GUEDES; VICARI, 2011) deixou possível a especificação dos papéis referentes a cada um dos tipos de agentes especialistas, sendo eles: Problem Solving (PS), Users and Agents Modelling (UAM) e Social Mediated Interactions (SMI).

- **InternalUseCase:** assim como a metaclasse *Actor*, na UML a metaclasse *UseCase* semanticamente descreve que os casos de uso representam requisitos externos, ou seja, funcionalidades que podem ser solicitadas por atores externos. Todavia, como os agentes realizam ações internas no sistema, que na imensa maioria das vezes não são sequer percebidas pelos usuários externos, (GUEDES; VICARI, 2011) derivou da metaclasse *BehavioredClassifier* uma nova metaclasse para representar os requisitos internos dos agentes.

- **Perception e Action:** essas metaclasses representam os casos de uso internos que contenham os passos necessários para um agente perceber ou realizar uma ação;

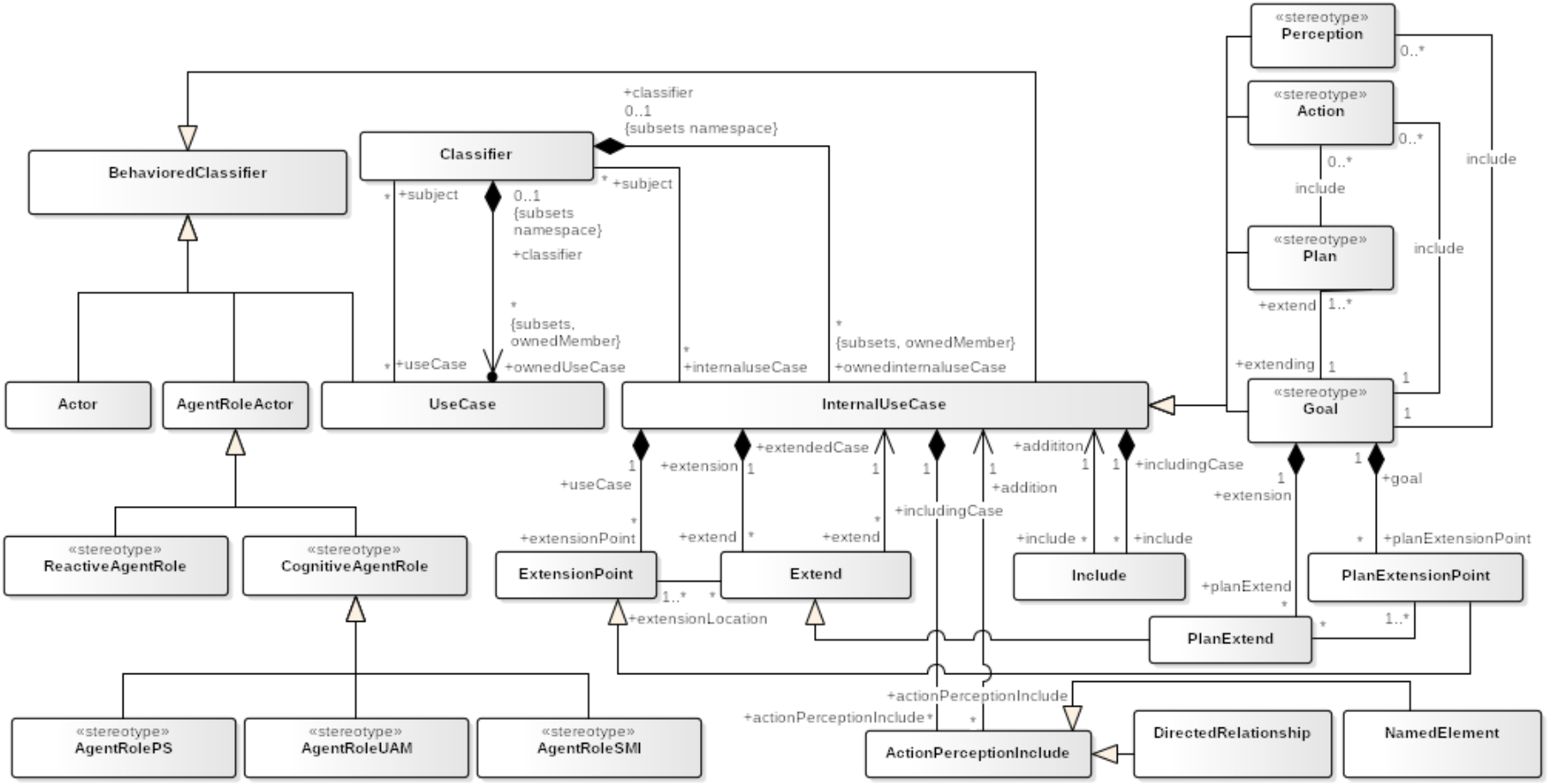
- **Goal:** essa metaclasse representa os objetivos de um agente contendo uma descrição de um desejo a ser atingido por um agente e as possíveis condições para que o desejo vire uma intenção;

- **Plan:** essa metaclasse representa um ou mais planos associados aos objetivos para que as percepções e condições necessárias para que um objetivo vire uma intenção.

Além das metaclasses estendidas e criadas citadas acima, outras metaclasses para representar a inclusão e extensão de comportamentos foram acrescidas, sendo elas:

- **Plan Extend:** representa por meio de uma associação de composição que um *Goal* pode ter muitos *Plan Extend* associados a ele. No entando um *Plan Extend* só pode estar associado a um único *Goal*.
- **Plan Extension Point:** estabelece uma diferença entre pontos de extensão de plano e pontos de extensão normais.
- **IncludeActionPerception:** estabelece diferente da (OMG, 2017) que a inclusão de uma Action ou Perception não obrigatoriamente inclui o comportamento de um caso de uso incluído. Assim retirando a recomendação de que ela um *include* seja usado quando há comportamentos comuns a mais de um caso de uso.

Figura 4: Metademolo de Guedes e Vicari (2011).



Fonte: (GUEDES; VICARI, 2011)

2.6 Mapeamento Sistemático

Um *Mapeamento Sistemático da Literatura (MSL)* tem como objetivo realizar uma busca ampla em estudos relacionados com um tópico específico, identificando as contribuições existentes para uma determinada área, tendo assim uma visão ampla sobre o tópico em questão (PETERSEN et al., 2008). No qual, um mapeamento pode ser conduzido utilizando estratégias de busca, sendo elas a busca em (I) bases de busca por meio de palavras-chave e/ou por meio da estratégia buscando através das (II) citações e/ou referências de um conjunto de inicial de estudos.

2.6.1 Busca por palavras-chave

Esse método de busca se baseia na elaboração de uma *string* de busca, que de acordo com Nakagawa et al. (2017), a *string* é uma combinação de palavras-chave com operadores lógicos, que tem como intuito ser utilizada para encontrar estudos nas fontes de busca, que por sua vez são locais onde as buscas são feitas para identificar os estudos primários relacionados ao tema sendo investigado.

2.6.2 *Snowballing*

A técnica de *snowballing* possui o objetivo de buscar estudos primários relacionados a um tópico de pesquisa especifico. Mas diferente da busca em bases de busca, o *snowballing* de acordo com Wohlin (2014), refere-se ao uso das referências e citações de um conjunto de artigos primários para a identificação de novos estudos primários.

Umas das principais vantagens dessa técnica é que ela não se limita apenas aos artigos *indexados* em bases bibliográficas. Outra diferença é que essa técnica não está sujeita a sensibilidade dos termos de uma *string* de busca. Isso possibilita a descoberta de estudos adicionais que podem não ter sido encontrados pelas *strings* de busca.

Além disso, essa técnica não se limita apenas a ser utilizada sozinha. De acordo com Petersen, Vakkalanka e Kuzniarz (2015), uma pesquisa pode obter melhores resultados ao combinar estratégias com base em palavras-chave com a análise das citações e referências. Desse modo reduzindo os problemas encontrados utilizando ambas as técnicas (NAKAGAWA et al., 2017).

Em linhas gerais a técnica é iniciada por meio da seleção de um conjunto inicial de estudos (sementes), em que os mesmos devem estar relacionados com o tópico a ser estudado. Após selecionar as sementes, inicia-se o processo de *snowballing*, que pode ser divido em duas estratégias:

- *Backward (reverso)*: Examina todos os estudos citados por um estudo no conjunto de estudos iniciais;

- *Forward (avante)*: Examina todos os estudos que fazem menção a um estudo do conjunto de estudos iniciais.

Essas estratégias podem ou não ser executadas em conjunto dentro de ciclos iterativos. Além disso, os ciclos de avaliação devem ser realizados até não haver mais novos estudos relevantes, definindo assim o fim do *snowballing* (NAKAGAWA et al., 2017).

2.7 Papyrus

Utilizou-se o *plugin* Eclipse de modelagem de código aberto *Papyrus* para estender o metamodelo de Guedes e Vicari (2011), aplicar as regras OCL sobre o mesmo e criar modelos a partir da instanciação do metamodelo estendido. Em linhas gerais, essa ferramenta fornece um ambiente integrado para a edição de modelos UML e SysML. Além disso, de acordo com Amaral (2017), uma das suas principais características é a compatibilidade com UML 2.5, servindo como um editor gráfico para a UML 2 definida pela OMG (2017), implementando 100% de sua especificação.

2.8 Object Constraint Language

Em linhas gerais, a Object Constraint Language (OCL) é um linguagem de modelagem que permite a especificação de detalhes de um modelo, permitindo adicionar restrições e regras específicas (KLEPPE et al., 2003). Além disso, para Ali et al. (2011), essa linguagem permite que os modeladores definam restrições em diferentes níveis de abstração e para diferentes tipos de modelos.

De acordo com Egea e Dania (2019), a OCL foi criada para modelar propriedades que não podiam ser facilmente capturadas ou representadas por meio da notação gráfica da UML, como por exemplo, anotações invariantes de uma classe. Nesse sentindo, a linguagem OCL se propõe a oferecer expressões livres de ambiguidade, além de buscar complementar os modelos da UML.

2.9 Grupo Focal

De acordo com Santos et al. (2016), a técnica de pesquisa grupo focal é formulada numa abordagem qualitativa. Essa técnica é utilizada para a coleta e analisa dados por meio das interações pessoais em forma de grupos que, são estimulados, por um pesquisador, a discutirem sobre um tema, assim permitindo aos entrevistados exporem suas ideias e estabelecerem opiniões sobre o tema em discussão.

Além disso, para Backes et al. (2011), o grupo focal pode ser utilizado para coleta e análise de dados. Nesse sentindo, há um deslocamento da posição do participante como porta-voz do grupo focal, passando a ser um sujeito ativo no processo analítico de determinado fenômeno. Contudo, quando o grupo focal é utilizado como técnica de

análise, há uma tendência a ocorrerem discussões que são incipientes e divergentes. Dessa forma, no intuito de mitigar esse problema, Backes et al. (2011) propõe como modelo de coleta e de análise de dados o uso da Análise SWOT.

A análise SWOT (*Strengths*, *Weaknesses*, *Oppotunities* e *Threats*), é uma técnica utilizada principalmente para a gestão e o planejamento em empresas. Em linhas gerais, essa técnica constitui-se no preenchimento de uma matriz, onde a mesma é dividida em quatro partes, em que cada parte devem ser preenchida com os conteúdos que representam o posicionamento da proposta em relação a cada um dos seguintes itens:

- Forças: tópicos fortes da proposta;
- Fraquezas: tópicos fracos da proposta;
- Oportunidades: tópicos que podem ser melhorados na proposta;
- Ameaças: tópicos que podem invalidar/desclassificar a proposta.

2.10 Lições do Capítulo

Neste capítulo foram apresentados conceitos importantes para o entendimento do estudo realizado. Em geral descrevemos o que é um sistema multiagente, bem como o que é um agente e suas principais características. Definimos também o que é um papel e como ele, ao ser interpretado por um agente, modifica o comportamento do mesmo. Detalhamos ainda que em algumas situações um agente pode sofrer com restrições temporais. Além disso, apresentamos conceitos relacionados a especificação de requisitos e como realizar a documentação dos mesmos. Nesse sentindo, também descrevemos a linguagem de modelagem UML, o diagrama de casos de uso e como ambos podem ser complementados com regras OCL.

3 TRABALHOS RELACIONADOS

Neste capítulo são apresentados o conjunto de trabalhos relacionados identificados por meio de um mapeamento sistemático complementado por um *snowballing*. Estes trabalhos, em linhas gerais, descrevem a representação de requisitos específicos para SMA por meio do DCU da UML adaptados ou não.

3.1 Escopo do Mapeamento Sistemático

Esse estudo teve como objetivo identificar as principais contribuições sobre a Engenharia de Requisitos (ER) para SMA. O principal objetivo foi identificar como é realizada a representação de requisitos para SMA. Além disso, foi avaliado também como literatura tem utilizado os DCU e suas extensões ou abordagens similares a eles para especificar os requisitos particulares para SMA.

3.2 Questões de Pesquisa

Com o intuito de atingir o objetivo desse MSL, duas questões de pequisa foram elaboradas. A primeira questão de pesquisa (RQ1) foi concebida para identificar como as abordagens para desenvolvimento de sistemas multiagentes abordam a representação de requisitos, em especial, os requisitos específicos para este tipo de sistema, como por exemplo, a identificação de agentes, os possíveis papéis que eles podem assumir, quais são seus objetivos, planos etc.

Já a segunda questão de pesquisa (RQ2) visa identificar como a literatura tem utilizado casos de uso, suas extensões ou abordagens similares a eles para especificar as requisitos particulares para SMA. As questões elaboradas para responder a essas dúvidas foram:

- **RQ1:** Como as abordagens de desenvolvimento de SMA suportam a representação de requisitos?

- **RQ2:** Considerando que as abordagens de desenvolvimento de SMA utilizem extensões dos casos de uso da UML para representar requisitos específicos para agentes, de que maneira isto é realizado?

3.3 *String* de busca

Para identificar os estudos relevantes para o nosso estudo, uma *string* de busca genérica (Figura 5) contendo as principais palavras-chave do escopo deste estudo foi elaborada. Também essa *string* sofreu ajustes para adaptar-se as particularidades de cada uma das bases bibliográficas. Além disso, é importante ressaltar que as *strings* utilizadas

nas bases *Engineering Village*, *Scopus* e *SpringerLink* possibilitaram o uso de um filtro para limitar as buscas a apenas estudos relacionados à área da ciência da computação.

Figura 5: String genérica.

```
(approach OR methodology OR process OR method OR technique OR profile OR
                        metamodel OR language)
                              AND
("requirements elicitation" OR "requirements analysis" OR "requirements
specification" OR "requirements validation" OR "requirements verification" OR
"requirements extraction" OR "requirements gathering" OR "requirements
                              modeling")
                              AND
("agent system" OR "multi-agent system" OR "multiagent system" OR MAS)
```

Fonte: O autor.

3.4 Bases de busca

Para realizar esse MSL, foram usadas apenas bases de dados que (i) possuem um mecanismo de busca baseado na web; (ii) tem um mecanismo de busca capaz de usar palavras-chave; e (iii) contém documentos da área da ciência da computação. Todas as bases utilizadas são apresentadas na Tabela 1.

Tabela 1: Bases bibliográficas utilizadas.

Base	Tipo	URL
ACM	Híbrida	http://dl.acm.org
Engineering Village	Motor de busca	http://engineeringvillage.com
IEEE Xplore	Bases bibliográficas	http://ieeexplore.ieee.org
ScienceDirect	Bases bibliográficas	http://www.sciencedirect.com
SpringerLink	Bases bibliográficas	http://link.springer.com
Scopus	Motor de busca	http://www.scopus.com

Fonte: O autor.

3.5 Critérios de Inclusão e Exclusão

Para determinar quais estudos poderiam ser incluídos nesse MSL, um critério de inclusão foi elaborado. O mesmo busca incluir estudos que realizam de alguma forma a representação de requisitos específicos para SMA, sendo ele:

- O estudo propõe a representação de requisitos específicos para o SMA.

Para excluir os estudos não relacionados às questões de pesquisa, os seguintes critérios de exclusão foram elaborados:

- Estudo não escrito em inglês.

- Estudo não fornece acesso total ao seu conteúdo.
- Estudo duplicado.
- As contribuições do estudo limitam-se apenas a exemplos de uso;

3.6 Extração dos dados

Para extrair as informações relevantes para responder às questões de pesquisa definidas na Seção 3.2, os seguintes critérios de extração foram definidos:

- Título (questão 1 e 2);
- Autor (questão 1 e 2);
- Forma como os requisitos são representados *e.g Casos de uso, Ontologia.* (questão 1);
- Forma como foi realizada a extensão do caso de uso da UML (questão 2).

3.7 Condução nas Bases de Busca

Essa busca[1] foi realizada entre o período de Fevereiro/2019 à Abril/2019. No qual os passos foram seguidos conforme definido na Seção 2.6, em que todas as etapas foram executadas usando a estratégia de *peer-review*.

Inicialmente, um estudo piloto foi executado para verificar a qualidade da *string* genérica e do protocolo definido. Neste estudo piloto, uma versão anterior da *string* de busca genérica, definida em conjunto com um especialista, foi executada na base bibliográfica *Scopus*. A execução retornou 116 estudos sobre os quais foram aplicados os critérios de exclusão e inclusão do protocolo, o que reduziu os estudos a apenas quatro.

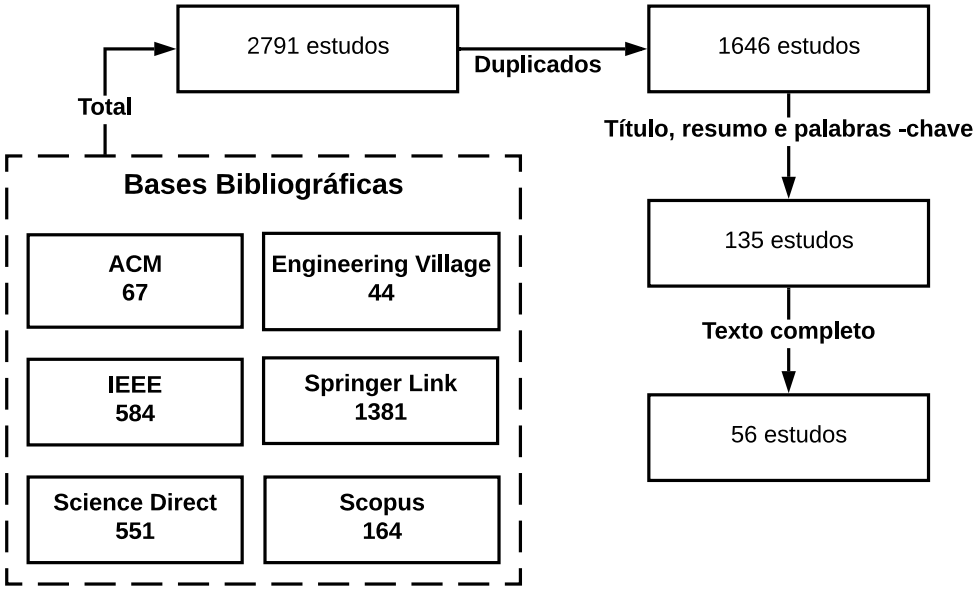
Devido ao fato de que poucos estudos foram identificados através da *string* do estudo piloto, a mesma sofreu modificações, buscando aumentar sua abrangência. Após se identificar que, além dos estudos relevantes anteriormente selecionados, novos estudos relevantes estavam sendo *indexados* pela nova *string*, esta passou a ser a *string* oficial do estudo.

Também adotamos duas fases para realizar a avaliação dos artigos: (I) leitura do título, resumo e palavras-chave das obras selecionadas principalmente, identificando os trabalhos promissores e eliminando os demais e (II) leitura completa dos trabalhos selecionados na primeira fase. Após a realização dessas etapas a extração dos dados foi realizada e as questões da pesquisa respondidas.

[1] Os artefatos da condução do mapeamento estão disponíveis em: https://drive.google.com/open?id=1k-pzQZd68K2yxMAl43IHJ9yWKCFfNock

No total foram identificados 2791 artigos extraídos de 6 bases de dados. Na etapa 2, após a identificação dos artigos, 1145 artigos duplicados foram removidos e 1646 artigos foram mantidos. Na etapa 3, os revisores analisaram o título, o resumo e as palavras-chave do artigo e os resultados apontam para 135 artigos mantidos e 1010 artigos excluídos. A análise do texto completo selecionou 56 artigos e excluiu 79 artigos (Etapa 4), e finalmente na Etapa 5 a extração de dados foi realizada nos 56 artigos remanescentes. Uma representação completa do processo de condução é mostrada na Figura 6.

Figura 6: Resumo das etapas de condução com os estudos restantes em cada etapa.



Fonte: O autor.

3.8 Condução do Snowballing

Entre o período de Agosto/2019 à Setembro/2019 esse *snowballing* foi conduzindo, em que o primeiro passo foi definir um conjunto de estudos sementes. Essa busca ocorreu através do *Google Scholar*, que de acordo com Nakagawa et al. (2017), é uma base de busca que realiza buscas multidisciplinares por trabalhos revisados, dissertações de mestrado teses de mestrado, livros artigos, entre outros. Assim se encaixando nos *Guidelines* descritos por (WOHLIN, 2014), em que deve-se buscar a diversidade, buscando abranger várias editoras diferentes, publicações de diferentes anos e autores.

A busca no *Google Scholar*, ocorreu por meio da elaboração de 4 *strings*, que podem ser vistas na Figura 7. Para as 3 primeiras foram consideradas as 3 primeiras páginas para cada *string* (30 resultados p/ cada). Já para a quarta e última foram consideradas as primeiras 10 páginas (100 resultados), totalizando 190 resultados.

Além disso, para o processo de seleção foram criados dois novos critérios de seleção (um de inclusão e um de exclusão). Isso ocorreu devido ao objetivo do *snowballing* ser

Figura 7: String usada no *Google Scholar*.

```
String 1 - modeling requirements agent OR "multi agent system"
String 2 - specification requirements agent OR "multi agent system"
String 3 - representation requirements agent OR "multi agent system"
String 4 - extension OR extend "use case"UML requirements agent OR "multi agent
system"
```

Fonte: O autor.

focado em apenas complementar os resultados obtidos relacionadas às extensões dos DCU para a representação requisitos específicos para SMA. Diferente do MSL, que buscou além dos DCU, identificar as formas de representação dos requisitos para SMA.

- Critério de inclusão - O estudo menciona representação de requisitos específicos para agentes em seu resumo/título/palavras-chave.
- Critério de exclusão - O estudo não apresenta uma extensão dos casos de uso da UML.

Após analisar cada um dos resultados das buscas no *Google Scholar*, em que o protocolo descrito na seção 2.6, em conjunto com os 2 novos critérios, foi seguido, foram encontrados 4 novos artigos. Além disso, os estudos identificados foram reunidos com os resultados do MSL, totalizando 8 artigos semente.

Após a definição das sementes o processo de seleção do *snowballing* foi executado e 1.070 artigos foram avaliados. No final da execução foram identificados e incluídos mais 6 artigos, totalizando assim 14 estudos. O processo e as incidências em cada um dos ciclos do *snowballing* pode ser visto na Tabela 2.

Tabela 2: Ciclos do *snowballing*.

Iteração	**Avaliados**	**Selecionados**
Ciclo 0	367	5
Ciclo 1	683	1
Ciclo 2	20	0

Fonte: O autor.

3.9 Discussão dos Resultados

Inicialmente, foi identificado que a maior parte das representações de requisitos é realizada por meio de casos de uso ou por mecanismos assemelhados (quando o conceito de casos de uso é estendido ou uma aplicação semelhante é proposta). A partir dessa descoberta, concluímos que os conceitos tradicionais da engenharia de requisitos vêm sendo aplicados na representação de requisitos para o desenvolvimento de SMA.

Também se verificou que a maioria das representações são gráficas, como pode ser observado na Tabela 3. Analisando esta tabela, podemos perceber que a representação de requisitos por meio de casos de uso ou similares é a representação gráfica mais utilizada. A segunda representação mais utilizada é a representação *Goal Model* com 18 ocorrências e em terceiro lugar a representação do *Role Model*. Observando essa tabela também é possível perceber que muitos estudos representam requisitos de forma textual, uma vez que temos 12 ocorrências representando requisitos funcionais e não funcionais textualmente e 11 ocorrências fazem uso da representação formal dos requisitos.

Tabela 3: Quantidade de ocorrências por tipo de representação.

Tipo de representação	**Ocorrências**	**Tipo de representação**	**Ocorrências**
Caso de uso e similares*	19	Notação i*	7
Modelo de objetivo	18	Diagrama de atividades	7
Modelo de papéis	16	Cenários	5
NF e/ou F Textual*	12	Ontologia	4
Descrição formal	11	Modelo de organização	4
Modelo de tarefa	10	Modelo de ambiente	2
Modelo de domínio	9	User Story	1

Fonte: O autor.

3.10 Utilização de extensões de casos de uso para representação de requisitos para SMA

Inicialmente, quando essa questão de pesquisa foi elaborada, tínhamos ciência de que a mesma poderia não ser respondida. Com esse estudo, além de buscar como os requisitos para SMA são representados, havia também um interesse especial na representação de requisitos funcionais específicos para SMA por meio de casos de uso ou similares. Com a realização do estudo piloto, a existência de estudos que realizavam justamente esse tipo de representação foi identificada. Assim foi possível perceber que essa questão poderia ser respondida e, portanto, a mesma foi mantida.

Dos estudos identificados no MSL, 10 estudos foram analisados para responder a essa pergunta. Como é possível observar na Tabela 4, de uma forma geral, as propostas estenderam a UML para resolver problemas específicos, como por exemplo a representação de agentes móveis. Posto isso, fica latente que essas soluções focam em linhas de pesquisa distintas e não foram identificados esforços para unificar e/ou complementar essas propostas em si.

Além disso, algumas propostas se destacam, como por exemplo o estudo de Saleh e El-Morr (2004) que, em sua extensão da UML, possibilitou a representação de agentes móveis, criando o ator móvel e o caso de uso móvel, representados no exemplo do autor

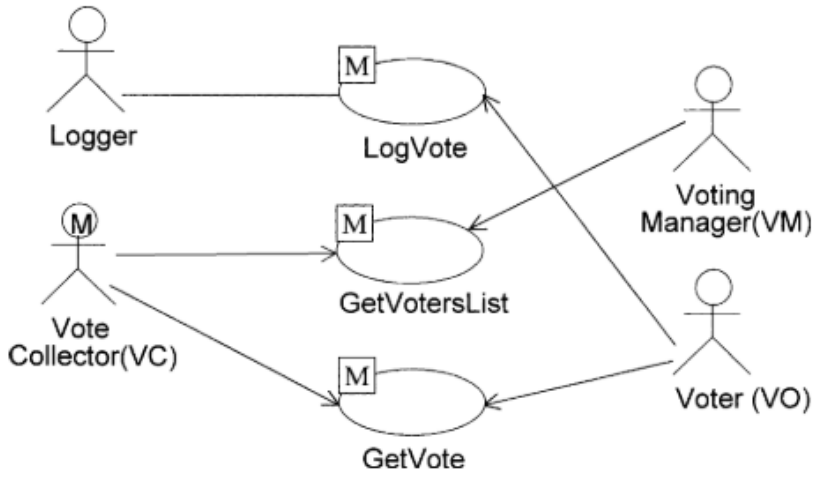
Tabela 4: Extensões dos DCU da UML para representar requisitos específicos de agentes.

	Agente	**CasoUso Interno**	**Agente-M**	**CasoUso-M**	**Agente-RT**	**CasoUso-RT**	**Agent-O**	**CasoUso-O**	**Objetivo**	**Crença**	**Plano**	**Percepção**	**Ação**	**Documentação**	**Arquiteturas**
(IGLESIAS et al., 1997)	X	X													
(HEINZE; PAPASIMEON; GOSS, 2000)	X	X												X	
(DEPKE; HECKEL; KÜSTER, 2002)	X	X							X						
(FLAKE; GEIGER; KÜSTER, 2001)	X	X							X			X			
(SALEH; EL-MORR, 2004)	X	X	X	X											
(SPANOUDAKIS; MORAITIS, 2008)	X	X													
(HAMIDANE; MOKHATI, 2010)	X	X													
(LAOUADI; MOKHATI, 2010)	X	X			X	X									
(GUEDES; VICARI, 2011)	X	X							X	X	X	X	X	X	X
(LAOUADI; MOKHATI, 2013)	X	X			X	X	X	X							

Fonte: O autor.

na Figura 8. O ator estendido busca modelar a interação do agente móvel com o sistema enquanto estiver em uma plataforma distante daquela em que o agente foi criado. Já o caso de uso móvel, representa um requisito funcional que envolve um ou mais atores móveis. Contudo, a proposta não aborda o modelo BDI, limitando-se apenas na representação das funcionalidades executadas pelos agentes.

Figura 8: Representação de um caso de uso móvel.

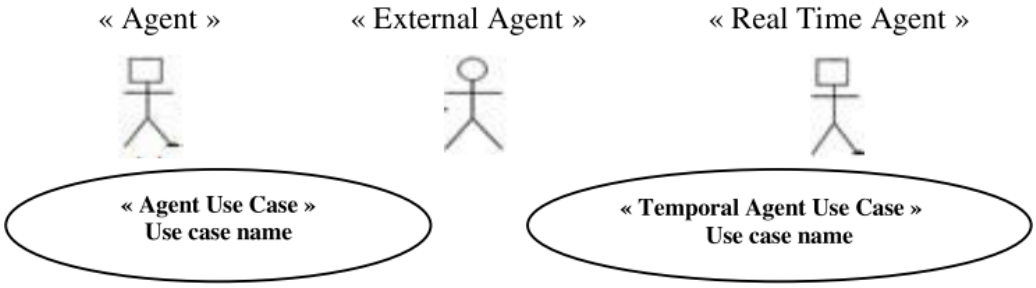


Fonte: Saleh e El-Morr (2004)

Para a representação de ART, (LAOUADI; MOKHATI, 2010) realizou a extensão da Agent UML (uma extensão da UML para agentes) para, através da aplicação de novos estereótipos nas metaclasses do DCU, ser possível representar agentes e ART, representados na Figura 9. No entanto, apesar de ter acrescido novas características à linguagem

com sua extensão, ainda se faz necessário estender a Agent UML para representar características especificas de agentes, tais como objetivos, percepções, ações, planos. Além de que, assim como os demais estudos, apenas houve a alterações semântica dos elementos, o que novamente ao olhar desse estudo, é uma prática incorreta.

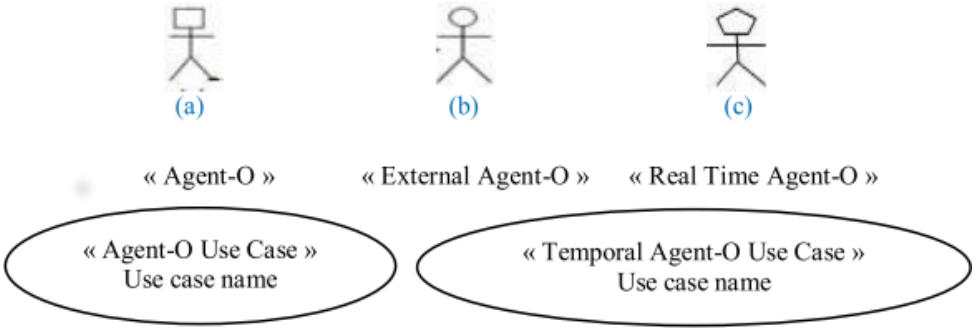
Figura 9: Representação dos elementos do DCU para ART.



Fonte: Laouadi e Mokhati (2010).

Em outra proposta Laouadi e Mokhati (2013), criaram 5 novos esteriótipos, sendo eles: «Agent-O Use Case» e «Temporal Agent-O Use Case», que são usados, respectivamente, para representar casos de uso que modelam funcionalidades executadas por agentes e casos de uso que modelam comportamentos executados por ART em algumas organizações. Já os elementos «Agent-O», «External Agent-O», «Real Time Agent-O» que descrevem, nessa ordem, os agentes dentro de uma organização, agentes externos à organização e ART que são *Agent-O*. A representação dos esteriótipos criados são representados na Figura 10.

Figura 10: Representação dos elementos do DCU para agentes móveis.



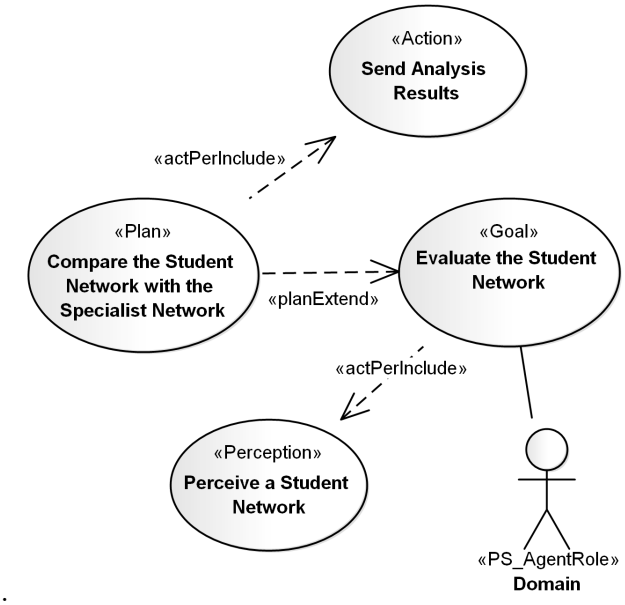
Fonte: Laouadi e Mokhati (2013).

Como observado, a maioria dos estudos apenas buscou representar os comportamentos dos agentes em um nível macro, resumido todas as particulariedades apenas a um caso de uso. Com exceção do estudo de Guedes e Vicari (2011), em que, na sua extensão do diagrama de casos de uso da UML, criou novas metaclasses e estereótipos, além de um modelo de documentação. A partir disso é possível representar papéis de agente, os

objetivos a eles associados, os planos para atingi-los, as percepções necessárias para que um objetivo se torne uma intenção e as ações que um agente assumindo um papel pode executar.

Em linhas gerais os papéis de agente são modelados como *AgentRoleActors* e os conceitos de objetivos, planos, percepções e ações como casos de uso internos contendo estereótipos *Goal*, *Plan*, *Perception* e *Action* para identificá-los. Um exemplo de aplicação dessa extensão é apresentado na Figura 11. Contudo, apesar de suportar o modelo BDI, o metamodelo ainda necessita do desenvolvimento de novas metaclasses para repreentar as funcionalidades relacionadas aos papéis de ART.

Figura 11: Representação de aplicação do metamodelo de Guedes e Vicari (2011)



Fonte: Guedes e Vicari (2011)

Além disso, com a exceção dos estudos de Flake, Geiger e Küster (2001) e Depke, Heckel e Küster (2002), nessa ordem, que criaram um estereótipo e um elemento para representar os objetivos de um agente Os demais estudos se limitaram apenas em representar pura e simplemeste os agentes e casos de uso internos, sem representar qualquer tipo de comportamente específico.

Outro destaque, são os estudos de Iglesias et al. (1997), Flake, Geiger e Küster (2001), Depke, Heckel e Küster (2002), Laouadi e Mokhati (2010) e Hamidane e Mokhati (2010), que criaram um elemento em comum para representar um agente em suas extensões da UML. Esse novo elemento, diferente dos atores em caso de uso (bonecos com a cabeça redonda), são bonecos com a cabeça quadrada. Assim apresentado uma diferença visual, além da diferença semântica, para representar agentes como internos ao sistema.

3.11 Ameaças à validade

As ameaças ao resultado do MSL, foram identificadas e categorizadas de acordo com Wohlin et al. (2012): validade de construto, validade interna, validade externa e validade de conclusão.

Validade de Construto: Uma ameaça de construção é a possível exclusão de estudos relevantes. Para mitigar esse problema, realizamos análises por pares, nas quais cada uma das avaliações agiu de forma independente durante o processo de condução e avaliação dos estudos..

Validade Interna: Possíveis ameaças podem ter surgido através do uso de métodos de pesquisa incorretos, o que pode levar à exclusão de estudos relevantes. Para mitigar isso, buscamos avaliar o protocolo com antecedência para verificar se os estudos relevantes estavam sendo incluídos na pesquisa. Além da interação constante com um especialista para a avaliação da direção MSL. Nesse sentindo, também aplicamos duas técnicas para encontrar estudos relacionados ao nosso objetivo, sendo a busca por bases de busca complementada por um snowballing.

Validade Externa: Questões externas, como a indisponibilidade de estudos, foram resolvidas com a pesquisa de iniciativas como o Google Scholar ou o contato direto com os autores.

Validade da Conclusão: Ameaças ao MSL podem surgir dos pesquisadores que o conduziram. Devido ao fato, de não terem conhecimento de seu próprio viés durante a condução e extração desse estudo. Assim, isso pode impactar em uma possível imprecisão na extração de dados, ameaçando a conclusão do estudo. Para mitigar essas ameaças, um especialista na área foi consultado durante todo o MSL. Além de realizar todas as etapas da condução, desde a seleção dos estudos até a extração dos dados por no mínimo dois pesquisadores independentes e, em caso de divergência, um especialista da área foi consultado.

4 EXTENSÃO DO METAMODELO

Neste capítulo são apresentado a extensão realizada do metamodelo de Guedes e Vicari (2011). Em linhas gerais, são discutidos e apresentados aspectos relacionados desde a extensão do metamodelo até a instanciação do mesmo na ferramenta Papyrus, no qual é utilizada para modelar um SMA-RT.

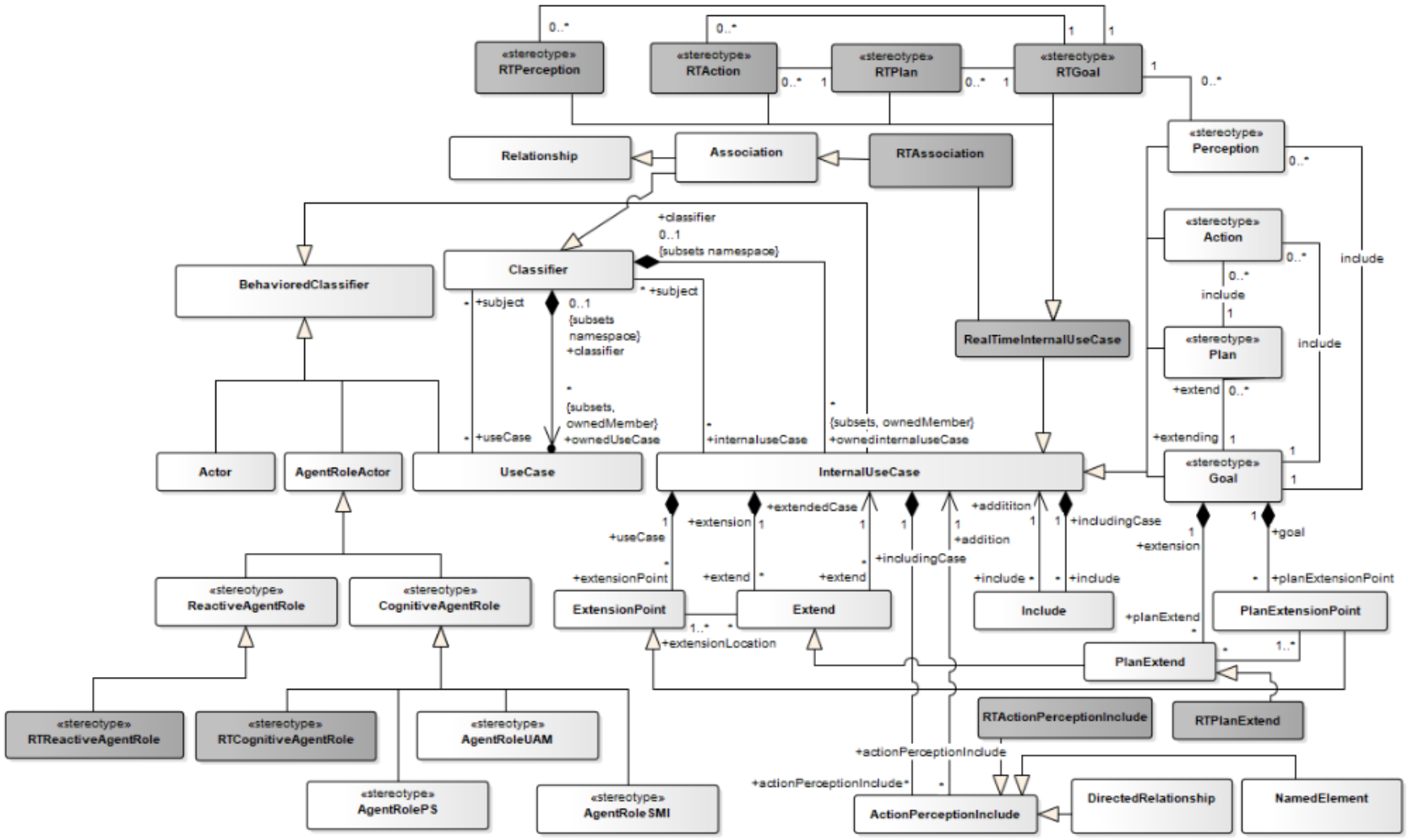
Como identificado, por meio dos trabalhos relacionados discutidos na seção 3, alguns estudos ao longo dos anos estenderam o DCU da UML de forma a adaptá-lo a domínios para os quais ele não possuía mecanismos, como por exemplo SMA.

Após uma metanálise, o estudo de Guedes e Vicari (2011) demostrou ser o mais completo. O referido metamodelo em comparação aos trabalhos relacionados, foi o único a se preocupar em representar comportamentos particulares de agentes, como objetivos, planos, ações e percepções seguindo o modelo BDI. Além disso, o estudo também foi o único a adicionar novas metaclasses e estereótipos em sua extensão do metamodelo da UML.

Contudo, a extensão de Guedes e Vicari (2011) ainda possui oportunidades de melhoria, como por exemplo a representação das características de ART. Dessa forma, identificamos a possibilidade de estender o referido metamodelo, de maneira a permitir a representação de funcionalidades internas de agentes que estivessem sujeitas a regras temporais.

Dessa forma, especializamos as metaclasses existentes no metamodelo de Guedes e Vicari (2011), para representar funcionalidades com restrições temporais criando 10 novas metaclasses (ilustradas na Figura 12). Essas novas metaclasses foram criadas com base nas principais características definidas na literatura sobre ART identificadas através de um *snowballing*, no qual o protocolo e seus resultados são apresentados no Apêndice A).

Figura 12: Extensão do Metamodelo de Guedes e Vicari (2011)



Fonte: O autor.

4.1 Estereótipos RTReactiveRole e RTCognitiveRole

Em relação a representação de papéis de agentes, o metamodelo de Guedes e Vicari (2011) permite representar papéis de agentes reativos ou cognitivos. Como descrito anteriormente, os papéis de agentes reativos apenas reagem a eventos, enquanto os papéis cognitivos possuem crenças, desejos e intenções sobre o ambiente em que se encontram. Nesse sentido, um agente também pode interpretar um papel de tempo real, sendo o mesmo reativo ou cognitivo. Ao assumir esse tipo de papel, o agente se compromete a atingir seus objetivos dentro de um intervalo determinado de tempo.

Todavia, atualmente o metamodelo de Guedes e Vicari (2011) não suporta a representação de papéis de agente de tempo real. Sendo assim, estendemos as metaclasses *ReactiveAgentRole* e *CognitiveAgentRole* do metamodelo de Guedes e Vicari (2011), criando duas novas metaclasses (ilustradas na Figura 12): *RTReactiveRole* e *RTCognitiveRole* e atribuindo-lhes o esterótipo *"stereotype"*. Dessa forma, essas novas metaclasses permitem aplicar os estereótipos `«RTReactiveRole»` e `«RTCognitiveRole»` em papéis de agentes reativos ou cognitivos, destacando assim que um agente ao interpretar um desses papéis deverá obrigatoriamente executar funcionalidades internas sujeitas a restrições temporais.

4.2 Metaclasse RTInternalUseCase

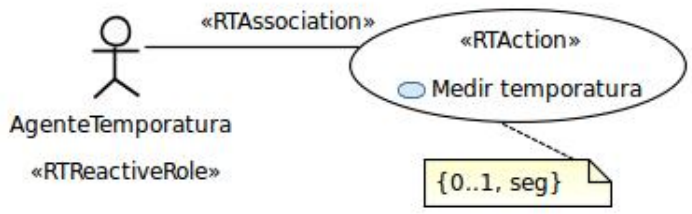
Uma vez que o metamodelo de Guedes e Vicari (2011) não possui nenhuma metaclasse capaz de representar funcionalidades internas sujeitas a restrições temporais, estendeu-se uma nova metaclasse denominada *RealTimeInternalUseCase* a partir da meclasse *InternalUseCase*. Essa nova metaclasse preliminarmente continha os *tagged values* TMin e TMax para identificar os tempos mínimo e máximo para que um comportamento fosse concluído.

Inicialmente, se pensou em estabelecer a unidade de tempo padrão como sendo milissegundos, todavia, durante um exercício de validação, percebeu-se que, em situações que se fez necessário representar outras unidade de tempo, os valores expressos ficaram extensos, o que comprometeu a usabilidade da representação. Dessa forma optou-se por criar um novo *tagged value* denominado de unidade de tempo (TUnit), definido em conjunto com os *tagged values* TMin e TMax, para expressar a unidade de tempo aplicada a cada comportamento. A Figura 13 apresenta um caso de uso interno em tempo real exposto a restrições temporais.

Além disso, três restrições OCL foram criadas para garantir que (I) os valores TMin, TMax sejam maiores do que 0 (Restrição 4.1); (II) o valor de TMin deve ser menor ou igual ao valor de TMax (Restrição 4.2).

Restrição 4.1: Regra OCL para impedir que os *tagged values* TMin TMax sejam maiores do que 0.

Figura 13: Caso de um interno em tempo real com uma restrição temporal.



Fonte: O autor.

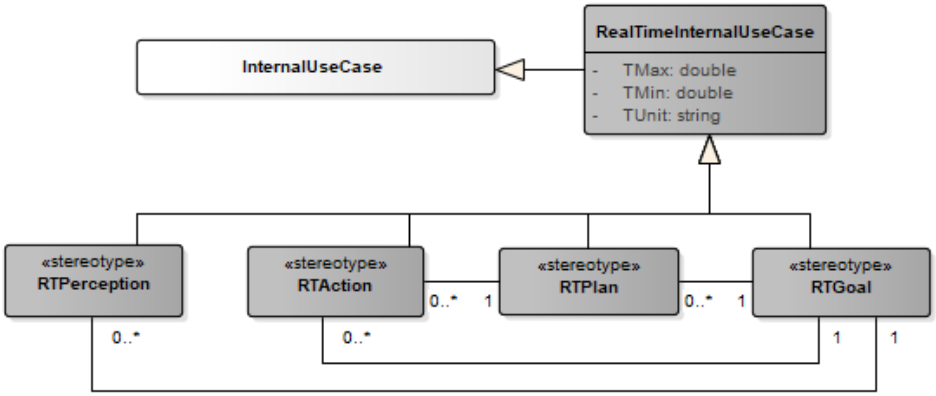
```
1 self.TMin >=0 and self.TMax >= 0
```

Restrição 4.2: Regra OCL para definir que o valor do *tagged value* Tmin precisa ser menor ou igual ao valor do *tagged value* TMax.

```
1 self.TMin <= self.TMax
```

Para representar restrições temporais em nível de modelo, utilizamos notas explicativas contendo restrições entre colchetes, estabelecendo o tempo mínimo, máximo e a unidade de tempo utilizada, como ilustrado na Figura 14.

Figura 14: Novas metaclasses extendidas a partir da metaclasse *InternalUseCase*.



Fonte: O autor.

4.3 Novos Estereótipos Comportamentais

No metamodelo de Guedes e Vicari (2011), os comportamentos de um agente são represetandos pela metaclasse InternalUseCase. Essa metaclasse é especializada pelas metaclasses `«Goal»`, `«Perception»`, `«Plan»` e `«Action»`, que são utilizadas como estereótipos de InternalUseCase. Contudo, nenhum desses estereótipos possibilita a representação de comportamentos sujeitos a restrições temporais. Sendo assim, optamos por especializar

a metaclasse *RealTimeInternalUseCase* por meio de quatro novas metaclasses para representar objetivos, percepções, ações e planos que obrigatoriamente precisem obedecer regras temporais. Estas metaclasses receberam o estereótipo "stereotype", dessa forma elas agem como estereótipos de *RealTimeInternalUseCase*.

4.3.0.1 Estereótipo «GoalRT»

Em Guedes e Vicari (2011), a metaclasse *Goal* representa a descrição de um objetivo e as possíveis condições para que esse objetivo seja alcançado. Porém, como descrito anteriormente, no contexto de ART os objetivos podem conter restrições temporais. Nesse sentindo o estereótipo «RTGoal» foi criado para permitir representar objetivos de tempo real, ou seja, objetivos que devem ser concluídos dentro de um determinado intervalo de tempo, representado por meio de retrições temporais. Um objetivo de tempo real pode estar associado a:

- planos em tempo real;
- ações em tempo real;
- percepções em tempo real;
- percepções.

É possível também associar um caso de uso interno com o estereótipo «Perception» a um objetivo em tempo real. Isso se torna possível devido ao fato de uma percepção poder ou não estar exposta a uma restrição temporal para acionar o "gatilho" para que um objetivo se torne uma intenção.

4.3.0.2 Estereótipo «RTPlan»

Como visto anteriormente, um papel de agente pode possuir planos para atingir seus objetivos. Esse conceito não muda na agência em tempo real, no qual o comportamento de um objetivo em tempo real pode ser estendido pelo comportamento de um ou mais planos em tempo real. Nesse sentido, a metaclasse *RTPlan* foi estendida a partir da metaclasse *RealTimeInternalUseCase* para representar esse conceito. Além disso, um *RTPlan* pode estar associado a (I) um ou mais objetivos e (II) nenhuma ou muitas ações. Além disso, como especificada na restrição 4.3, a soma de todos os *tagged values* de plano em tempo real, quando associado a um comportamento com esteriótipo «RTGoal», não podem exceder o tempo total do seu objetivo em tempo real.

Restrição 4.3: Regra OCL para definir que o tempo total de um plano em tempo real não pode exceder o tempo total do seu objetivo em tempo real.

```
1 self.goal.TMin - self.goal.TMax >= self.TMin - self.TMax
```

4.3.0.3 Estereótipo «RTPerception»

As percepções de um agente atuam como auxiliares para determinar quando um objetivo pode ou não tornar-se uma intenção. Nesse sentindo, uma percepção pode precisar ocorrer dentro de um período de tempo especifico, caso contrário, ela não será realmente útil. Sendo assim, criamos a metaclasse *RTPerception* para representar as percepções em tempo real. Esse tipo de comportamento, quando modelado, deve estar associada obrigatoriamente a um objetivo em tempo real.

4.3.0.4 Estereótipo «RTAction»

Assim como os demais comportamentos já descritos, as ações de um agente também podem estar expostas a restrições temporais. Dessa forma, a metaclasse *RTAction* foi criada para representar ações que devem ser executadas dentro de um período de tempo específico. Estas ações em tempo real precisam estar associadas a um objetivo em tempo real ou a um plano em tempo real e como especificado na restrição 4.4, o tempo total de uma ação em tempo real não pode exceder o tempo total do seu objetivo/plano em tempo real.

Restrição 4.4: Regra OCL para definir que o tempo total de uma ação em tempo real não pode exceder o tempo total do seu objetivo/plano em tempo real.

```
1 -- Associacao com um RTPlan
2 self.plan.TMax - self.plan.TMin >= self.TMax - self.TMin
3
4 -- Associacao com um RTGoal
5 self.goal.TMax - self.goal.TMin >= self.TMax - self.TMin
```

4.3.1 Associação *RTPlanExtend*

Como descrito anteriormente, para que um objetivo em tempo real seja atingido, deve haver pelo menos um plano em tempo real para executá-lo, ficando evidente uma relação entre as metaclasses *RTGoal* e *RTPlan*, em que os planos em tempo real estão fortemente associados aos objetivos de um papel de agente.

No metamodelo que está sendo estendido, existe a metaclasse *PlanExtend*, que associa metaclasse *Goal* com a metaclasse *Plan*. Portanto, em nosso contexto de comportamentos em tempo real, optamos por estender a associação *PlanExtend* por meio da metaclasse *RTPlanExtend*.

Dessa forma, como apresentado na Restrição 4.5, uma associação *RTPlanExtend* torna claro que a extensão se refere especificamente a uma extensão de um objetivo em tempo real pelo comportamento de um plano em tempo real. Sendo assim, somente planos em tempo real podem estar associados a objetivos de tempo real, nunca outro tipo de plano.

Restrição 4.5: Regras OCL para restringir a associação entre *RTGoal* e *RTPlan*.

```
-- SOURCE
self.base_Extend.source.getAppliedStereotypes()->exists(
        stereotype | stereotype . name = 'RTPlan'

-- TARGET
self.base_Extend.target.getAppliedStereotypes()->exists(
        stereotype | stereotype . name = 'RTGoal')
```

4.3.2 Associação *RTInclude ap*

Para que um objetivo se torne uma intenção, é necessário que exista uma condição. Dessa forma, quando o agente percebe que essa condição foi atendida, o mesmo passa a realizar ações para atingir seu objetivo. Sendo assim, para representar esses conceitos, Guedes e Vicari (2011) em seu metamodelo, derivaram a metaclasse *ActionPerceptionInclude* a partir da metaclasse *Include*.

Como já abordado anteriormente, os conceitos de um agente normal também podem ser aplicados a ART. Dessa forma, extendemos a metaclasse *RTActionPerceptionInclude*, a partir da *ActionPerceptionInclude*, para representar a inclusão de ações e percepções em tempo real em um objetivo em tempo real.

Assim, ao modelar um objetivo em tempo real, associamos uma ou mais ações/-percepções em tempo real a um objetivo de tempo real por meio de a associação *RTActionPerceptionInclude*. Isso significa que, quando o comportamento com o estereótipo *RTGoal* for executado por um papel de agente, os comportamentos com os estereótipos *RTAction* e *RTPerception* também serão executados.

4.3.3 Associação *RTAssociation*

Como descrito anteriormente, apenas papéis de ART podem estar associados a comportamentos em tempo real. Dessa forma, estendemos a associação *RTAssociation* a partir da metaclasse *Association*. Dessa forma, com essa novo relacionamento é possível restringir as associações apenas entre meclasses com estereótipos de tempo real. Além disso, para que essa restrição seja respeitada, duas regras OCL foram criadas (Restrição 4.6).

Restrição 4.6: Regras OCL para restringir associações apenas entre estereótipos de tempo real.

```
-- SOURCE
self.base_Association.endType.getAppliedStereotypes()->exists(
    stereotype | stereotype.name = 'RTReactiveRole' or
        stereotype.name = 'RTCongnitiveRole')
```

```
5
6  -- TARGET
7  self.base_Association.endType.getAppliedStereotypes()->exists(
8      stereotype | stereotype.name = 'RTGoal' or
9      stereotype.name = 'RTPlan' or stereotype.name =
10     'RTAction' or stereotype.name = 'RTPerception')
```

4.4 Exemplo de Aplicação do Metamodelo Estendido

Com o intuito de verificar a adequabilidade da extensão realizada, especificamos um SMA contendo papeis de ART, inspirado no ambiente apresentado por (DIPIPPO et al., 2001). O referido ambiente possui requisitos com restrições temporais se destacam principalmente no papel de agente AgenteCompraEVenda, posto que, devido a volatilidade desse tipo de operação, esse papel deve atingir seus objetivos quase que instantaneamente, ressaltando que em caso de resultado contrário, o objetivo não será atingido e terá como consequência despesas vultuosas ao usuário. Além disso, a especificação pode ser observada na Figura 15.

4.4.1 Requisitos

Para essa representação, tomamos como base o problema descrito por DiPippo et al. (2001), em que realizamos algumas adaptações. O autor descreve um ambiente em que vários ART são responsáveis por coordenar e realizar recomendações inteligentes relacionadas a compra e venda de ações na bolsa de valores. Em linhas gerais, no ambiente se encontram quatro tipos diferentes de agentes: um `AgenteUsuario`, um `AgenteCotacao`, um `AgenteTendencia` e um `AgenteCompraEVenda`.

O `AgenteUsuario` se comunica com o usuário humano para determinar seus requisitos, como a quantidade de dinheiro a ser gasto e as preferências do setor de mercado. O `AgenteUsuario` também se comunica com os outros agentes no sistema para poder fazer recomendações de compra e venda ao Usuário.

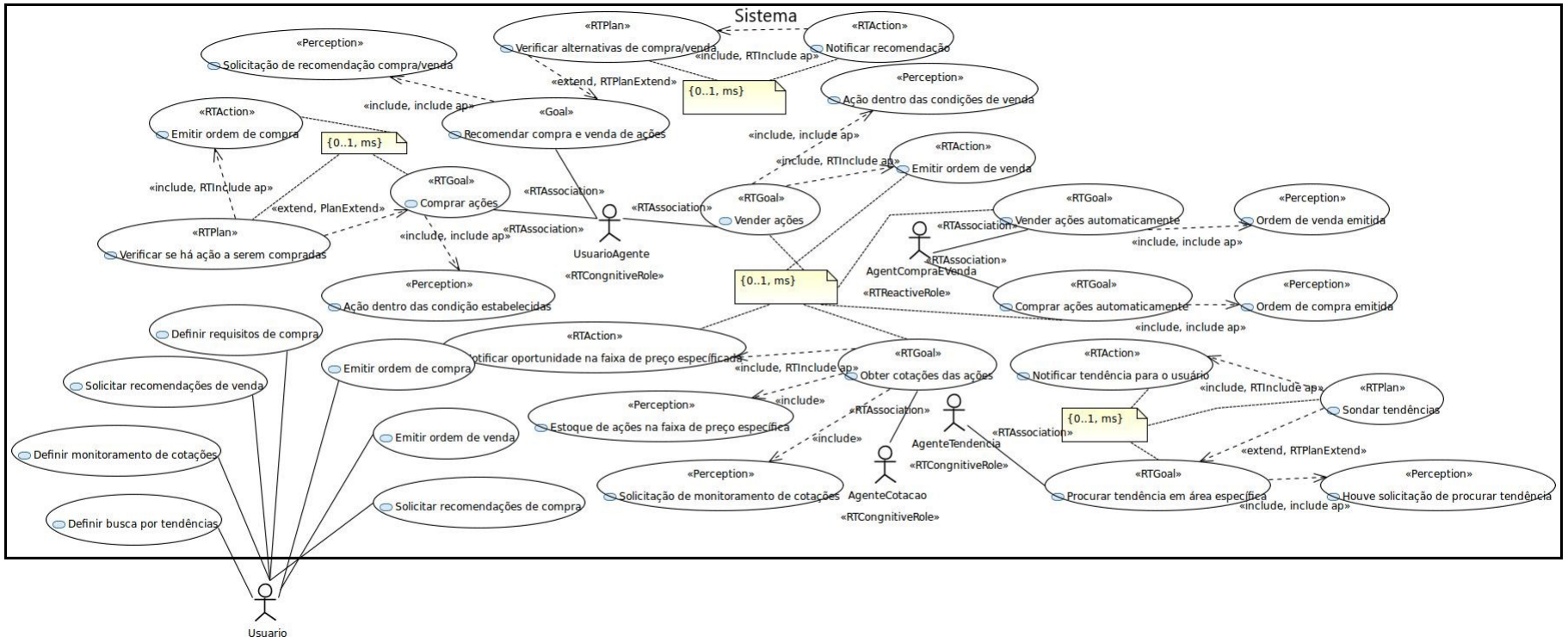
Cada `AgenteCotacao` tem a capacidade de obter cotações de ações em um setor específico do mercado. Ele também pode monitorar um estoque específico para uma faixa de preço específica.

Um `AgenteTendencia` procura tendências específicas do mercado, em que cada `AgenteTendencia` pode ser responsável por um determinado tipo de tendência, como por exemplo um aumento nos estoques do setor de biotecnologia.

O `AgenteCompraEVenda` é responsável pelas compras e vendas de ações, em que esse agente pode agir de forma autônoma se o usuário humano tiver expressado ao `AgenteUsuario` que as transações podem ser feitas automaticamente.

Se o usuário quiser se envolver em cada transação, o `AgenteUsuario` poderá fazer a recomendação ao usuário e notificar o `AgenteCompraEVenda` para realizar a transação. Sendo assim, nesse contexto as restrições de tempo neste SMA derivam da volatilidade dos preços no mercado de ações. Por exemplo, se o `AgenteUsuario` determinar que o usuário deseja comprar 100 ações do estoque de uma empresa qualquer, o usuário deve especificar um prazo de tempo para o `AgenteCompraEVenda` realize as transações desejadas. Além disso, todas as ações dos agentes possuem uma restrições de tempo de até 0,1 segundos.

Figura 15: Casos de uso internos sobre o sistema compra/venda de ações.



Fonte: O autor.

4.5 Lições do Capítulo

Nesse capítulo apresentamos como a extensão do metamodelo de Guedes e Vicari (2011) foi realizada para suportar a representação de funcionalidades em tempo real. Além disso, por meio de um exemplo de usa extensão foi possível identificar a adequabilidade no nosso metamodelo em relação a representação de requisitos para ART e seus comportamentos. Além disso, o metamodelo foi instanciado na ferramenta Papyrus, no qual é possível modelar SMA-RT utilizando o metamodelo estendido, assim como validar os modelos gerados a partir de regras OCL.

5 AVALIAÇÃO DA PROPOSTA

Este capítulo tem por finalidade apresentar o planejamento e condução de um grupo focal para avaliar a extensão apresentada nesse estudo. Além disso, os resultados obtidos por meio do grupo focal são expostos e discutidos, assim como as ameaças a validade do mesmo.

5.1 Planejamento do grupo focal

O grupo focal foi planejado para ser realizado com alunos de mestrado do Programa de Pós-Graduação em Engenharia de Software matriculados na disciplina de Engenharia de Software Orientada a Agentes.

Inicialmente um termo de consentimento livre (disponível no Apêndice B.1) e esclarecido foi desenvolvido para ser assinado pelos alunos dispostos a participar do grupo focal. Após isso, aos alunos que concordarem e assinarem o termo de consentimento, devem ser apresentados conceitos descrevendo em linhas gerais o que é um grupo focal e como o mesmo é conduzido (apresentação disponível no Apêndice C).

Após isso, devem ser apresentados conceitos relacionados a agentes em tempo real, bem como a extensão do metamodelo. Nessa etapa é necessário assegurar que o conhecimento passado durante a apresentação seja suficiente para que cada participante do grupo tenha conhecimento mínimo para discutir sobre o assunto em pauta.

Depois da apresentação, deve ser permitido um tempo (no máximo 1 hora) para que os participantes apliquem e discutam sobre os conceitos apresentados anteriormente. Essa atividade poderá ser realizada em grupo ou individualmente. Além disso, será permitido o debate e a consulta entre os demais participantes do grupo. O principal objetivo é fazer com que os participantes usem e reflitam sobre o uso/aplicação da extensão do metamodelo.

Após a atividade prática, os participantes devem responder individualmente um questionário com perguntas abertas e fechadas. Finalmente, os participantes devem preencher em conjunto uma matriz usando a técnica de análise SWOT.

5.2 Execução do grupo focal

O grupo focal ocorreu no dia 6/11/2019 e contou com 6 participantes. O primeiro passo na condução foi distribuir o termo de consentimento. Nesse momento se destacou o objetivo do grupo focal, além de que os alunos poderiam retirar seu consentimento ou interromper a participação a qualquer momento, sem sofrer qualquer tipo de penalidade ou prejuízo. Além disso, também se destacou que, ao participar do grupo focal, os participantes não teriam nenhum custo e nem receberiam qualquer vantagem financeira ou acadêmica.

A etapa de assinatura dos termos demorou cerca de 5 minutos e todos os alunos presentes aceitaram participar, totalizando 6 participantes. Após isso, iniciou-se a segunda etapa, na qual houve uma breve explicação sobre o que era um grupo focal e como o mesmo iria ser conduzido. Já na terceira etapa, os participantes foram expostos a conceitos relacionados a agentes em tempo real e ao metamodelo estendido. É importante ressaltar que durante a sessão não houve dúvidas sobre os assuntos apresentados.

Na quarta etapa, um SMA em tempo real foi apresentado e explicado em detalhes em conjunto com a sua representação utilizando o metamodelo estendido. Após a explicação, um instrumento contendo a descrição de um SMA em tempo real foi fornecido aos participantes. Nesse momento, os alunos se juntaram em 3 grupos (2, 3 e 1 alunos) para modelarem em conjunto a solução.

Em linhas gerais, as principais discussões ficaram em torno de como abstrair cada um dos comportamentos e suas restrições de tempo analisando os requisitos fornecidos. O tempo demorado para realizar a modelagem foi de 35 minutos a 1 hora. Além disso, ao passo que cada grupo finalizava a sua modelagem, os mesmos foram respondendo seus questionários (enumerados abaixo) e após o último grupo entregar sua representação, iniciou-se uma discussão aberta para realizar o preenchimento da matriz SWOT.

1. A representação de requisitos não-funcionais é importante em um documento de especificação de requisitos.
2. A representação de requisitos não-funcionais de tempo são importantes em um documento de especificação de requisitos para sistemas multiagentes de tempo real.
3. O metamodelo expressa adequadamente os conceitos que ele representa.
4. O metamodelo cobre os conceitos mínimos necessários para a modelagem de requisitos para sistemas multiagentes de tempo real.
5. Quais dificuldades você encontrou em utilizar o metamodelo?
6. Você teria alguma sugestão de melhoria para o metamodelo? Qual(is)?

5.3 Resultados dos questionários

Analisando a resposta dos participantes, é possível observar, a representação de requisitos não-funcionais é importante em um documento de especificação de requisitos, em que 50% concordou totalmente e 50% concordou com essa afirmação.

Em relação à segunda afirmação 33,33% dos participantes concordaram totalmente, enquanto 66,66% concordaram. Já na terceira questão 16,77% dos participantes concordaram totalmente, enquanto 83,33% concordaram. Por fim, em relação à terceira afirmação, 16,77% dos participantes concordaram totalmente, enquanto 83,33% concordaram.

Já para a questão ("Quais dificuldades você encontrou em utilizar o metamodelo?"), um participante reportou que os diagramas ficaram muito extensos por conta da grande quantidade de elementos. Também nesse sentindo, outro participante afirmou que as representações a partir do metamodelo ficam poluídos visualmente, devido a grande quantidade de elementos; Também houve um relato em relação a representação das restrições temporais, em que para representar grandes números de tempo, a unidade de tempo milissegundos dificulta a representação.

Em relação a sexta questão ("Você teria alguma sugestão de melhoria para o metamodelo? Qual(is)?"), houve sugestões em relação a representação das unidades temporais. Assim, dois participantes descreveram que uma melhoria poderia ser feita em relação a representação de várias unidades de tempo, como por exemplo minutos e/ou segundos (estava limitado apenas a milissegundos).

5.3.1 Análise SWOT

Através da técnica de análise SWOT foi possível identificar as forças, fraquezas, oportunidades e ameaças. Como pode ser observado na Figura 16, as principais forças estão na facilidade de entender o metamodelo e sua aplicação, além de possuir uma motivação clara e ser relevante para representar as noções de agência de tempo real, possibilitando aplicar as restrições já na etapa de engenharia de requisitos.

Já as fraquezas da extensão são a grande quantidade de elementos criados para especificar um SMA de tempo real. Isso acaba afetando a usabilidade de proposta, já que para cada um dos casos de uso internos é necessário especificar um estereótipo, além de especificar a restrição de tempo. Nesse sentindo, foram encontradas oportunidades de melhoria, em que definir elementos para cada um dos comportamentos dos agentes, por exemplo ao invés de ser uma elipse com um estereótipo «RTGoal» para representar um objetivo em tempo real, usar formas geométricas simples, como um quadrado ou representar os papéis de agentes com atores de cabeça quadrada. Porém, para isso seria necessário desenvolver uma ferramenta CASE.

Também como oportunidade foi destacada a padronização das unidades de medida. Nesse sentindo, no momento do grupo focal, a medida padrão definida para todas as restrições temporais foi milissegundos. Durante o grupo focal a atividade prática havia uma funcionalidade que deveria ser executada em minutos, o que nessa situação representar em milissegundos ficou impraticável. Assim, criamos um novo atributo na metaclasse *RealTimeInternalUseCase* denominado de unidade de tempo (TUnit), no qual o mesmo deve ser definido em conjunto com os atributos TMin e TMax, em que TUnit pode expressar qualquer unidade de tempo.

Figura 16: Análise SWOT gerada a partir do grupo focal.

Forças

1. Fácil de entender;
2. Motivação clara;
3. Relevante para representar agentes de tempo real;
4. Possibilita o uso de restrições temporais nos comportamentos dos agentes.

Fraquezas

1. Diagrama extenso quando tem muitos casos de uso;
2. Usabilidade ruim por conta de muitos estereótipos;
3. Números grandes ficam ruins de representar em milissegundos.

Oportunidades

1. Padronizar as unidades de medida;
2. Definir novos elementos para cada comportamento;
3. Usar atores com cabeça quadrada.

Ameaças

1. Não houve ameaças.

Fonte: O autor.

5.4 Ameaças à Validade

Para que uma avaliação obtenha um resultado válido, é necessário analisar as ameaças que podem comprometer a sua validade. Dessa forma, abaixo são listadas as ameaças a avaliação identificadas e quais foram as ações tomadas para mitigá las. Wohlin et al. (2012), categoriza as ameaças em 4 tipos: validade de construto, validade interna, validade externa e validade de conclusão.

- Validade da conclusão: O perfil dos participantes do grupo focal é uma ameaça significativa a validade dos resultados da avaliação, pois a avaliação foi realizada apenas com um pequeno grupo de alunos de mestrado.
- Validade interna: Os conceitos apresetados durante a apresentação dos conceitos relacionados agentes de tempo real e metamodelo estendido pode não ter ficado totalmente claro aos participantes.
- Validade externa: Os participantes selecionados para o experimento podem não representar um grupo significativo para a área de estudo. Para mitigar essa ameaça, foram selecionados alunos de mestrado em Engenharia de Software que estavam cursando uma disciplina relacionada a sistemas multiagentes.
- Validade de construto: A extensão foi utilizada pelos participantes em apenas um problema prático, dessa forma é possível que os participantes não puderam ex-

trair todos os recursos fornecidos pelo metamodelo estendido. Também fato de estar sendo avaliado pode provocar um certo nervosismo nos participantes, principalmente por serem acadêmicos. Dessa forma, para mitigar isso, os participantes foram informados de que os mesmos não estavam sendo avaliados e que não teriam prejuízos ou favorecimentos na disciplina que estavam cursando.

6 CONSIDERAÇÕES FINAIS

Este trabalho demonstra a extensão de um metamodelo de agência para a representação de agentes de tempo real e suas restrições temporais. Ainda durante este trabalho conduzimos um mapeamento sistemático para identificar como os requisitos para SMA eram representados e identificar como as extensões dos DCU são utilizados para especificar os requisitos particulares para SMA. Além disso, identificamos as principais noções de agência de tempo real através de um *snowballing*, por meio da qual definimos o que são agentes e sistemas multiagentes de tempo real,

Nesse MSL identificamos que o metamodelo de Guedes e Vicari (2011) atualmente é o mais completo em relação à especificação de requisitos funcionais específicos para SMAs, no entanto este metamodelo apresentava limitações em relação a especificação de requisitos relacionados a agentes de tempo real. Dessa forma, realizamos a extensão do referido metamodelo, que baseou-se nas definições de agentes de tempo real identificadas anteriormente. Nessa extensão novas metaclasses e estereótipos foram criados, a fim de contemplar a representação de requisitos de agentes expostos a restrições temporais. Em linhas gerais, restrições de tempo foram adicionadas aos conceitos relacionados aos objetivos, planos, ações e percepções dos papéis de agentes apresentados no metamodelo de Guedes e Vicari (2011).

Por fim, a extensão foi avaliada por meio de um grupo focal, que teve como participantes alunos matriculados no programa de pós-graduação em Engenharia de Software. Além disso, destacamos os esforços realizados para submeter esses resultados no Empirical Software Engineering International (ESEM) e na Escola de Engenharia de Software (ERES).

A principal limitação desse trabalho está em não haver uma ferramenta de modelagem que suporte a representação dos elementos criados nessa extensão, tendo sido necessário instanciar o metamodelo na ferramenta *Papyrus*. Contudo, acreditamos que isso não inviabiliza a utilização do metamodelo, uma vez que essa abordagem facilita e possibilita a tarefa de especificar requisitos específicos para agentes de tempo real. Dessa forma, tornando mais expressivo e de fácil entendimento os requisitos desse tipo particular de sistema.

Como trabalhos futuros pretende-se criar uma ferramenta de modelagem para suportar a representação dos elementos do metamodelo proposto, visando aumentar a produtividade de nossa especificação. Além disso, ainda há (I) a possibilidade de desenvolver um documento de especificação de requisitos voltado para requisitos de sistemas multiagentes, incluindo requisitos de tempo real; (II) estender o metamodelo para agentes organizacionais e móveis, aumentando o suporte do metamodelo para representar requisitos de outros tipos de agentes; (III) incluir a extensão a um processo de engenharia de requisitos para definir passo a passo como aplicá-la; (IV) pretendemos explorar a modelagem conceitual para sistemas multiagentes em tempo real, a fim de capturar os conceitos

necessários a esses sistemas, representando as informações necessárias para os agentes planos e objetivos em tempo real; (V) explorar o diagrama de temporização e verificar o quão útil ele poderia ser para detalhar as funcionalidades de um agente de tempo real.

REFERÊNCIAS

ALI, S. et al. A search-based ocl constraint solver for model-based test data generation. In: IEEE. **2011 11th International Conference on Quality Software**. Madri, Espanha: IEEE, 2011. p. 41–50. Citado na página 41.

AMARAL, E. F. **Uma abordagem para especificação de requisitos sensíveis ao contexto baseado em casos de uso para sistemas autoadaptativos**. Campus Alegrete, Alegrete: Universidade Federal do Pampa, 2017. Trabalho de Conclusão do Curso. <http://dspace.unipampa.edu.br:8080/jspui/handle/riu/1938>. (Acessado em 10/11/2019). Citado na página 41.

ASHAMALLA, A. N. N. **A requirement modelling framework for real-time multi-agent systems**. Tese (Doutorado) — Universidade de Tecnologia de Sydney. Faculdade de Engenharia e Tecnologia da Informação, 2017. Citado 2 vezes nas páginas 31 e 33.

BACKES, D. S. et al. Grupo focal como técnica de coleta e análise de dados em pesquisas qualitativas. **O mundo da saúde**, São Paulo, v. 35, n. 4, p. 438–42, 2011. Citado 2 vezes nas páginas 41 e 42.

BAJO, J. et al. An execution time planner for the artis agent architecture. **Engineering Applications of Artificial Intelligence**, Elsevier, v. 21, n. 5, p. 769–784, 2008. Citado 2 vezes nas páginas 31 e 33.

BATES, J. et al. The role of emotion in believable agents. **Communications of the ACM**, Citeseer, v. 37, n. 7, p. 122–125, 1994. Citado na página 29.

BERENBACH, B. et al. **Software & systems requirements engineering: in practice**. Nova York, Estados Unidos: McGraw-Hill, Inc., 2009. Citado 2 vezes nas páginas 33 e 34.

BOES, J.; MIGEON, F. Self-organizing multi-agent systems for the control of complex systems. **Journal of Systems and Software**, Elsevier, v. 134, p. 12–28, 2017. Citado na página 25.

BOOCH, G.; JACOBSON, J.; RUMBAUGH, J. **Uml - Guia do Usuário**. 2. ed. Brasil: Elsevier Brasil, 2016. ISBN 9788535285659. Citado na página 34.

BOURQUE, P.; FAIRLEY, R. E. et al. **Guide to the software engineering body of knowledge (SWEBOK (R))**. 3. ed. Los Alamitos, CA, USA: IEEE Computer Society Press, 2014. ISBN 9780769551661. Citado 3 vezes nas páginas 25, 33 e 34.

CARRASCOSA, C. et al. Real-time agents: Reaction vs. deliberation. In: EUROPEAN WORKSHOP ON MULTI-AGENT SYSTEMS, EUMAS04. **Agents in dynamic and real-time environments Workshop**. Barcelona, Espanha, 2004. p. 63–70. Citado na página 31.

CARRASCOSA, C. et al. Deliberative server for real-time agents. In: SPRINGER. **International Central and Eastern European Conference on Multi-Agent Systems**. Heidelberg, Berlin, 2003. p. 485–496. Citado na página 31.

CARRASCOSA, C. et al. Behaviour management for real time agents. **Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial**, v. 9, n. 25, p. 39–48, 2005. Citado na página 31.

CARRASCOSA, C. et al. A meta-reasoning model for hard real-time agents. In: SPRINGER. **Conference of the Spanish Association for Artificial Intelligence**. Heidelberg, Berlin, 2005. p. 42–51. Citado na página 31.

CHIN, K. O. et al. Agent architecture: An overview. **Transactions on science and technology**, v. 1, n. 1, p. 18–35, 2014. Citado 2 vezes nas páginas 29 e 30.

DEPKE, R.; HECKEL, R.; KÜSTER, J. M. Formal agent-oriented modeling with uml and graph transformation. **Science of Computer Programming**, Elsevier, v. 44, n. 2, p. 229–252, 2002. Citado 2 vezes nas páginas 49 e 51.

DICK, J.; HULL, E.; JACKSON, K. **Requirements engineering**. 3. ed. Londres, Reino Unido: Springer, 2017. ISBN 9781849964050. Citado na página 34.

DIPIPPO, L. C. et al. A real-time multi-agent system architecture for e-commerce applications. In: IEEE. **Proceedings 5th International Symposium on Autonomous Decentralized Systems**. Dallas, Texas, USA, 2001. p. 357–364. Citado 3 vezes nas páginas 31, 33 e 60.

DIPIPPO, L. C.; HODYS, E.; THURAISINGHAM, B. Towards a real-time agent architecture-a whitepaper. In: IEEE. **Proceedings. Fifth International Workshop on Object-Oriented Real-Time Dependable Systems**. Monterey, California, USA, 1999. p. 59–64. Citado na página 31.

DORRI, A.; KANHERE, S. S.; JURDAK, R. Multi-agent systems: A survey. **IEEE Access Volume 6**, IEEE, v. 6, p. 28573–28593, 2018. Citado na página 25.

DRAGONI, A. F.; SERNANI, P.; CALVARESI, D. When rationality entered time and became a real agent in a cyber-society. In: **Recent Trends and Applications in Computer Science and Information Technology**. Tirana, Albânia: Proceedings of the 3rd International Conference on Recent Trends and Applications in Computer Science and Information Technology, 2018. p. 167–171. Citado 3 vezes nas páginas 26, 31 e 32.

DUNIN-KEPLICZ, B.; VERBRUGGE, R. **Teamwork in multi-agent systems: A formal approach**. 1. ed. Reino Unido: John Wiley & Sons, 2011. ISBN 9780470699881. Citado na página 33.

EGEA, M.; DANIA, C. Sql-pl4ocl: an automatic code generator from ocl to sql procedural language. **Software & Systems Modeling**, Springer, v. 18, n. 1, p. 769–791, 2019. Citado na página 41.

FLAKE, S.; GEIGER, C.; KÜSTER, J. M. Towards uml-based analysis and design of multi-agent systems. **Symposium on Information Science Innovations in Engineering of Natural and Artificial Intelligent Systems**, Citeseer, Dubai, 2001. Citado 2 vezes nas páginas 49 e 51.

GALLIERS, J. R. **A theoretical framework for computer models of cooperative dialogue, acknowledging multi-agent conflict**. Tese (Doutorado) — Open University, Reino Unido, 1988. Citado 2 vezes nas páginas 29 e 30.

GUEDES, G. T. A. **UML 2 - Uma Abordagem Prática - 3a. Edição**. São Paulo: Novatec Editora, 2018. ISBN 9788575226469. Citado 2 vezes nas páginas 35 e 37.

GUEDES, G. T. A.; VICARI, R. **Um metamodelo UML para a modelagem de requisitos em projetos de sistemas multiagentes**. Tese (Doutorado) — Universidade Federal do Rio Grande do Sul - UFRGS, 2012. Citado 2 vezes nas páginas 34 e 35.

GUEDES, G. T. A.; VICARI, R. M. Applying a uml metamodel to the requirements modeling in multi-agents systems projects-the apa case study. 2011. Citado 18 vezes nas páginas 15, 25, 26, 36, 37, 39, 41, 49, 50, 51, 53, 54, 55, 56, 57, 59, 63 e 71.

HAMIDANE, F.; MOKHATI, F. Towards formalizing multi-agent systems functional requirements in maude. **International Journal of Advanced Research in Computer Science**, International Journal of Advanced Research in Computer Science, v. 1, n. 2, 2010. Citado 2 vezes nas páginas 49 e 51.

HEINZE, C.; PAPASIMEON, M.; GOSS, S. Specifying agent behaviour with use cases. In: SPRINGER. **Pacific Rim International Workshop on Multi-Agents**. Berlin, Heidelberg, 2000. p. 128–142. Citado na página 49.

IGLESIAS, C. A. et al. Analysis and design of multiagent systems using mas-commonkads. In: SPRINGER. **International Workshop on Agent Theories, Architectures, and Languages**. Berlin, Heidelberg, 1997. p. 313–327. Citado 2 vezes nas páginas 49 e 51.

JULIAN, V.; BOTTI, V. Developing real-time multi-agent systems. **Integrated Computer-Aided Engineering**, IOS Press, v. 11, n. 2, p. 136, 2004. Citado 2 vezes nas páginas 31 e 33.

JULIAN, V.; BOTTI, V. **Multi-Agent Systems**. Basiléia, Suíça: Multidisciplinary Digital Publishing Institute, 2019. Citado na página 25.

JULIAN, V. et al. Simba: an approach for real-time multi-agent systems. In: SPRINGER. **Catalonian Conference on Artificial Intelligence**. Berlin, Heidelberg, 2002. p. 282–293. Citado 2 vezes nas páginas 31 e 33.

KLEPPE, A. G. et al. **MDA explained: the model driven architecture: practice and promise**. Boston, Massachusetts: Addison-Wesley Professional, 2003. Citado na página 41.

KRÓL, D.; NOWAKOWSKI, F. Practical performance aspects of using real-time multi-agent platform in complex systems. In: IEEE. **2013 IEEE International Conference on Systems, Man, and Cybernetics**. Manchester, Reino Unido, 2013. p. 1121–1126. Citado na página 31.

LAOUADI, M. A.; MOKHATI, F. A novel formal specification approach for real time multi-agent system functional requirements. In: SPRINGER. **German Conference on Multiagent System Technologies**. Berlin, Heidelberg, 2010. p. 15–27. Citado 3 vezes nas páginas 49, 50 e 51.

LAOUADI, M. A.; MOKHATI, F. Towards an organizational model for real time multi-agent system specification. In: IEEE. **2013 Science and Information Conference**. Londres, Reino Unido, 2013. p. 577–584. Citado 2 vezes nas páginas 49 e 50.

NAKAGAWA, E. Y. et al. **Revisão sistemática da literatura em engenharia de software: Teoria e Prática**. São Paulo: Elsevier Brasil, 2017. Citado 3 vezes nas páginas 40, 41 e 46.

NAKAGAWA, H. et al. A framework for validating task assignment in multiagent systems using requirements importance. In: SPRINGER. **International Conference on Principles and Practice of Multi-Agent Systems**. Berlin, Heidelberg, 2010. p. 443–458. Citado na página 25.

OMG. **ABOUT THE UNIFIED MODELING LANGUAGE SPECIFICATION**. 2017. Disponível em: <https://www.omg.org/spec/UML/>. Citado 4 vezes nas páginas 34, 36, 38 e 41.

PATTEN, M. L.; NEWHART, M. **Understanding research methods: An overview of the essentials**. Nova York: Routledge, 2017. Citado na página 26.

PETERSEN, K. et al. Systematic mapping studies in software engineering. In: **12th International Conference on Evaluation and Assessment in Software Engineering**. Itália: BCS Learning & Development Ltd., 2008. p. 68–77. Citado na página 40.

PETERSEN, K.; VAKKALANKA, S.; KUZNIARZ, L. Guidelines for conducting systematic mapping studies in software engineering: An update. **Information and Software Technology**, Elsevier, v. 64, p. 1–18, 2015. Citado na página 40.

QASIM, A. **Formal Modelling of Real-Time Self-Adaptive Multi-Agent Systems**. Tese (Doutorado) — GC UNIVERSITY LAHORE, 2017. Citado 3 vezes nas páginas 25, 31 e 33.

QASIM, A.; KAZMI, S. A. R.; FAKHIR, I. Formal specification and verification of real-time multi-agent systems using timed-arc petri nets. **Adv. Elect. Comput. Eng.**, v. 15, n. 3, p. 73–78, 2015. Citado na página 31.

REGNELL, B. Requirements engineering with use cases–a basis for software development. **Department of Commnunication Systems, Lund Institute of Technology. Lund University, Sweden**, 1999. Citado 2 vezes nas páginas 33 e 34.

ROSENSCHEIN, J. S.; GENESERETH, M. R. Deals among rational agents. In: **Readings in Distributed Artificial Intelligence**. Los Angeles, California: Elsevier, 1988. p. 91. Citado na página 29.

RUSSELL, S. J.; NORVIG, P. **Artificial intelligence: a modern approach**. Malásia: Malaysia Pearson Education Limited, 2016. Citado 2 vezes nas páginas 25 e 29.

SALEH, K.; EL-MORR, C. M-uml: an extension to uml for the modeling of mobile agent-based software systems. **Information and Software Technology**, Elsevier, v. 46, n. 4, p. 219–227, 2004. Citado 2 vezes nas páginas 48 e 49.

SANTOS, R. C. da S. et al. O grupo focal como técnica de coleta de dados na pesquisa em educação: Aspectos éticos e epistemológicos. **Encontro Internacional de Formação de Professores e Fórum Permanente de Inovação Educacional**, v. 9, n. 1, 2016. Citado na página 41.

SHOHAM, Y. Agent-oriented programming. **Artificial intelligence**, Elsevier, v. 60, n. 1, p. 51–92, 1993. Citado na página 29.

SILVA, C. Separating crosscutting concerns in agent oriented detailed design: The social patterns case. **Centro de Informática da Universidade Federal de Pernambuco, Brasil**, 2007. Citado na página 32.

SILVA, V. T. D.; LUCENA, C. J. D. From a conceptual framework for agents and objects to a multi-agent system modeling language. **Autonomous Agents and Multi-Agent Systems**, Springer, Berlin, Heidelberg, v. 9, n. 1-2, p. 145–189, 2004. Citado na página 32.

SLHOUB, K.; CARVALHO, M.; BOND, W. Recommended practices for the specification of multi-agent systems requirements. In: IEEE. **2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)**. [S.l.], 2017. p. 179–185. Citado na página 25.

SOLER, J. et al. Towards a real-time multi-agent system architecture. **International Conference on Autonomous Agents and Multiagent Systems**, v. 2002, 2002. Citado 2 vezes nas páginas 31 e 33.

SOMMERVILLE, I. **Software engineering 9th Edition**. São Paulo, Brasil: Pearson, 2011. ISBN 9780137035151. Citado na página 33.

SPANOUDAKIS, N.; MORAITIS, P. The agent modeling language (amola). In: DOCHEV, D.; PISTORE, M.; TRAVERSO, P. (Ed.). **Artificial Intelligence: Methodology, Systems, and Applications**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. p. 32–44. ISBN 978-3-540-85776-1. Citado na página 49.

TRENCANSKY, I.; CERVENKA, R. Agent modeling language (aml): A comprehensive approach to modeling mas. **Informatica 29**, v. 29, n. 4, 2005. Citado 2 vezes nas páginas 25 e 32.

VICARI, R. M.; GLUZ, J. C. An intelligent tutoring system (its) view on aose. **International Journal of Agent-Oriented Software Engineering**, Inderscience Publishers, v. 1, n. 3-4, p. 295–333, 2007. Citado 3 vezes nas páginas 25, 30 e 37.

VIRUEL, M. L. R. Requirements modeling for multi-agent systems. 2011. Citado na página 25.

WOHLIN, C. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: CITESEER. **Proceedings of the 18th international conference on evaluation and assessment in software engineering**. Londres, Reino Unido, 2014. p. 38. Citado 2 vezes nas páginas 40 e 46.

WOHLIN, C. et al. **Experimentation in software engineering**. Berlin, Heidelberg: Springer Science & Business Media, 2012. Citado 2 vezes nas páginas 52 e 68.

WOOLDRIDGE, M. **An introduction to multiagent systems**. Reino Unido: John Wiley & Sons, 2009. ISBN 9780470519462. Citado 4 vezes nas páginas 25, 30, 32 e 33.

WOOLDRIDGE, M.; JENNINGS, N. R. Intelligent agents: Theory and practice. **The knowledge engineering review**, Cambridge University Press, v. 10, n. 2, p. 115–118, 1995. Citado 2 vezes nas páginas 25 e 29.

Apêndices

APÊNDICE A – SNOWBALLING SOBRE AGENTES EM TEMPO REAL

Protocol Snowballing Real-Time Agents

Lukas Gaedicke

[1]

1 Snowballing

De acordo com [Wohlin, 2014], a técnica de *snowballing* refere-se ao uso de referências e citações de artigos para identificar estudos adicionais e que não são indexados pelas bases de busca. Em linhas gerais a técnica é iniciada por meio da seleção de um conjunto de estudos inicias (sementes), que devem ser avaliados e aprovados pelo protocolo do *snowballing*. Além disso, o conteúdo dos estudos incluidos deve auxiliar a respoder às questões de pesquisa.

Após selecionar as sementes, inicia-se o processo de *snowballing*, que é dividido em dois passos e devem ser executados em ciclos iterativos até não houver a inclusão de novos estudos.

- Backward: Examina todos os estudos citados por um estudo da lista de estudos iniciais;
- Forward: Examina todos os estudos que citam um estudo da lista de estudos iniciais.

1.1 Questões de pesquisa

- O que é um agente de tempo real?
- O que é um sistema multiagente de tempo real?
- Quais comportamentos de um agente podem ser expostos às restrições temporais?

1.2 Processo de seleção

Para determinar quais estudos relevantes deveriam ser incluídos em nossa pesquisa, aplicamos critérios de inclusão e exclusão. Primeiramente definiram-se critérios de inclusão para servir como filtros de forma a incluir apenas estudos que descrevem agentes e/ou sistemas multiagentes em tempo real.

- IC01 - O estudo menciona agentes em tempo real em seu resumo/título/palavras-chave?
- IC02 - O estudo menciona sistemas multiagente em tempo real em seu resumo/título/palavras-chave?

Na segunda etapa de seleção, critérios de exclusão foram utilizados como um segundo filtro, em que buscou-se eliminar os estudos que não estavam relacionados ao contexto de agentes em tempo real. Dessa forma, os estudos que se encaixavam com alguma das questões a seguir foram excluidos.

- Estudo duplicado;
- Estudo escrito em outra língua que não seja inglês;
- O estudo não pode ser completamente acessado;
- O estudo não apresenta no texto completo uma definição sobre comportamentos/características de agentes em tempo real;
- O estudo faz referência a uma definição de um estudo já incluído.

1.3 Estratégia de extração de dados

Para extrair dados de forma a responder às questões de pesquisa da Seção 1.1, foram definidos 3 critérios, sendo eles:

- Definição de um agente em tempo real;
- Definição de um SMA em tempo real;
- Comportamentos de um agente expostos a restrições temporais.

2 Execução

O primeiro passo foi buscar um conjuto de estudos sementes. A busca ocorreu através do *Google Scholar*, que de acordo com [Nakagawa et al., 2017] é uma base de busca que realiza buscas multidiplicinares por trabalhos revisados, dissertações de mestrado teses de mestrado, livros artigos e entre outros. Assim se encaixando nos *Guidelines* descritos por [Wohlin, 2014], em que deve-se buscar a diversidade, buscando abranger várias editoras diferentes, publicações de diferentes anos e autores. Como resultado, foram obtidos 7 estudos sementes.

A partir disso, o processo de *snowballing* efetivamente foi realizado, em que foram analisados 1087 estudos. Do montante final, foram incluidos 8 estudos em 3 iterações, totalizando 15 estudos.

Table 1: Iterações do *snowballing*.

Iteração	Avaliados	Selecionados
Iteração 0	371	4
Iteração 1	582	1
Iteração 2	29	3
Iteração 3	97	0

3 Respondendo as questões

3.1 O que é um agente de tempo real?

Para [DiPippo et al., 1999] [Julian and Botti, 2004] [Król and Nowakowski, 2013] [Qasim, 2017], brevemente, um agente em tempo real é um agente cujas ações podem ser limitadas por restrições de tempo. Já para [Ashamalla, 2017], em uma definição mais elaborada, os ATRs são agentes que levam em consideração o tempo para atingir seus objetivos. Além de concordarem com as definições anteriores, os autores [Julian et al., 2002] [Soler et al., 2002] [Carrascosa et al., 2003] [Carrascosa et al., 2005a] [Carrascosa et al., 2004] descrevem que as restrições dos ARTs podem ser ”hard” e/ou ”soft”, e que essa classificação, segundo [Carrascosa et al., 2004], depende se os ARTs têm requisitos temporais criticos ou não. Também [Carrascosa et al., 2005b], reforça que, se um

ART com restrições de tempo apresentar resultados insatisfatórios, podem haver consequências catastróficas, e perante isso o agente é classificado como ”hard”.

Outra definição é apresentada por [DiPippo et al., 2001], que define que um ART deve atingir seus objetivos dentro de restrições de tempo específicas, mas que ao tentar atingir seus objetivos pode possivelmente impactar a qualidade dos seus resultados. Além disso, [Bajo et al., 2008] descreve que um ART deve garantir o cumprimento de suas restrições de tempo e, ao mesmo tempo, deve tentar atingir suas metas ou objetivos. Nesse sentindo, [Qasim et al., 2015] afirma que para esse tipo de agente cumprir seus objetivos dentro dos prazos específicos é importante, já que o oposto disso faz com eles percam sua usabilidade.

Finalmente para [Dragoni et al., 2018], um ART é um processo em execução cuja correção depende não apenas da totalidade de seu executável, mas também do momento em que a ação é ”executada”. Nesse sentido, um ART é um processo em execução que atua em um ambiente e que precisa ser executado a tempo, possivelmente raciocinando explicitamente sobre o tempo. Fato esse que se faz necessário incorporar a ele um relógio (*timer* ou contador), além de que se o mesmo raciocina sobre o tempo, provavelmente suas ações/planos irão mudar dependendo da situação, em função do tempo atual/tempo restante para atingir seus objetivos. Podendo assim também alterar seus planos com base no requisitos de qualidade de seus objetivos.

3.2 O que é um SMA de tempo real?

Para [Julian et al., 2002], [Soler et al., 2002], [Julian and Botti, 2004], [Bajo et al., 2008], [Qasim, 2017] um SMA em tempo real é um sistema multiagente com pelo menos um agente em tempo real. Já para [Ashamalla, 2017] sistemas multiagentes em tempo real são SMAs com restrições de tempo. Além disso, [DiPippo et al., 2001] descreve que no SMA-RT os agentes se comunicam, coordenam e negociam para atingir seus objetivos, dentro do prazo especificado e com restrições de qualidade. Bajo em [Bajo et al., 2008] acrescenta que, em um SMA-RT, as comunicações e protocolos entre agentes não podem ser muito complexos, já que podem impactar no desempenho dos agentes.

3.3 Quais comportamentos de um agente são expostos às restrições temporais?

Como pode ser observado na Tabela 2, a grande maioria dos comportamentos dos agentes expostos a restrições temporaís, descritos pelos estudos analisados, são os objetivos e as ações. Fato esse que ocorre devido a grande parte dos estudos não considerar o modelo BDI, assim descosiderando os demais comportamentos e interpretando que os agentes são apenas orietados a objetivos, e atingem os mesmos por meio de ações.

Também para [Soler et al., 2002], [Julian and Botti, 2004] e [Bajo et al., 2008] as responsabilidades de um agente são expostas às restrições de tempo. Contudo, as responsabilidades não são especificadas, ou seja, fica a critério do leitor interpretar o que é uma responsabilidade, que pode ser um plano, um objetivo ou uma percepção. Assim fornecendo uma oportunidade para aplicar restrições temporais não apenas nas ações e objetivos, como descrito pelos demais autores.

References

Ashamalla, A. N. N. (2017). *A requirement modelling framework for real-time multi-agent systems*. PhD thesis.

Bajo, J., Julián, V., Corchado, J. M., Carrascosa, C., de Paz, Y., Botti, V., and de Paz, J. F. (2008). An execution time planner for the artis agent architecture. *Engineering Applications of Artificial Intelligence*, 21(5):769–784.

Carrascosa, C., Fabregat, J., Terrasa, A., and Botti, V. (2004). Real-time agents: Reaction vs. deliberation. In *Agents in dynamic and real-time environments Workshop*, pages 63–70.

Carrascosa, C., Rebollo, M., Julián, V., and Botti, V. (2003). Deliberative server for real-time agents. In *International Central and Eastern European Conference on Multi-Agent Systems*, pages 485–496. Springer.

Carrascosa, C., Terrasa, A., Fabregat-Pinilla, J., and Botti, V. J. (2005a). Behaviour management for real time agents. *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial*, 9(25):39–48.

Carrascosa, C., Terrasa, A., García-Fornes, A., Espinosa, A., and Botti, V. (2005b). A meta-reasoning model for hard real-time agents. In *Conference of the Spanish Association for Artificial Intelligence*, pages 42–51. Springer.

DiPippo, L. C., Fay-Wolfe, V., Nair, L., Hodys, E., and Uvarov, O. (2001). A real-time multi-agent system architecture for e-commerce applications. In *Proceedings 5th International Symposium on Autonomous Decentralized Systems*, pages 357–364. IEEE.

DiPippo, L. C., Hodys, E., and Thuraisingham, B. (1999). Towards a real-time agent architecture-a whitepaper. In *Proceedings. Fifth International Workshop on Object-Oriented Real-Time Dependable Systems*, pages 59–64. IEEE.

Dragoni, A. F., Sernani, P., and Calvaresi, D. (2018). When rationality entered time and became a real agent in a cyber-society. In *RTA-CSIT*, pages 167–171.

Julian, V. and Botti, V. (2004). Developing real-time multi-agent systems. *Integrated Computer-Aided Engineering*, 11(2):135–149.

Julian, V., Carrascosa, C., Rebollo, M., Soler, J., and Botti, V. (2002). Simba: an approach for real-time multi-agent systems. In *Catalonian Conference on Artificial Intelligence*, pages 282–293. Springer.

Król, D. and Nowakowski, F. (2013). Practical performance aspects of using real-time multi-agent platform in complex systems. In *2013 IEEE International Conference on Systems, Man, and Cybernetics*, pages 1121–1126. IEEE.

Nakagawa, E. Y., Scannavino, K. R. F., Fabbri, S. C. P. F., and Ferrari, F. C. (2017). *Revisão sistemática da literatura em engenharia de software: Teoria e Prática*. Elsevier Brasil.

Qasim, A. (2017). *Formal Modelling of Real-Time Self-Adaptive Multi-Agent Systems*. PhD thesis, GC UNIVERSITY LAHORE.

Qasim, A., Kazmi, S. A. R., and Fakhir, I. (2015). Formal specification and verification of real-time multi-agent systems using timed-arc petri nets. *Adv. Elect. Comput. Eng.*, 15(3):73–78.

APÊNDICE B – ARTEFATOS DO GRUPO FOCAL

B.1 Termo de consentimento

Termo de Consentimento Livre e Esclarecido

Você está sendo convidado(a) para participar, como voluntário(a), de um estudo para avaliação de um metamodelo estendido para representar requisitos para sistemas multiagentes em tempo real. Este estudo é parte integrante da pesquisa chamada "Extensão de um metamodelo para a representação de requisitos com restrições temporais", coordenada pelo Prof. Dr. Gilleanes Thorwald Araujo Guedes.

Por meio deste documento, a qualquer momento, você poderá solicitar esclarecimentos adicionais sobre o estudo ou sobre a pesquisa. Também poderá retirar seu consentimento ou interromper a participação a qualquer momento, sem sofrer qualquer tipo de penalidade ou prejuízo. Ao participar deste estudo você não terá nenhum custo e nem receberá qualquer vantagem financeira. Seus dados pessoais serão mantidos em sigilo. Os resultados deste estudo serão armazenados pelo pesquisador responsável e poderão ser divulgados em publicações científicas.

Para participar deste estudo, você precisará: ouvir as explicações, ler o material fornecido, participar das discussões e reportar sugestões de melhorias e críticas. Ao participar deste estudo você corre o risco de frustrar-se por não conseguir realizar as atividades solicitadas. Esperamos que os resultados deste estudo contribuam para a minimização da complexidade da representação de requisitos de sistemas multiagentes em tempo real.

Após esses esclarecimentos, solicitamos o seu consentimento de forma livre para que participe deste estudo. Para tanto, preencha os itens a seguir.

Eu, _____, tendo em vista as informações acima apresentadas, de forma livre e esclarecida, aceito participar deste estudo.

_____, ____ de _____ de _____.

_____
Assinatura do Participante

B.2 Questionário

Questionário de Avaliação do Metamodelo

1. **A representação de requisitos não-funcionais é importante em um documento de especificação de requisitos.**

Discordo totalmente	Discordo	Neutro	Concordo	Concordo totalmente
☐	☐	☐	☐	☐

2. **A representação de requisitos não-funcionais de tempo são importantes em um documento de especificação de requisitos para sistemas multiagentes de tempo real.**

Discordo totalmente	Discordo	Neutro	Concordo	Concordo totalmente
☐	☐	☐	☐	☐

3. **O metamodelo expressa adequadamente os conceitos que ele representa.**

Discordo totalmente	Discordo	Neutro	Concordo	Concordo totalmente
☐	☐	☐	☐	☐

4. **O metamodelo cobre os conceitos mínimos necessários para a modelagem de requisitos para sistemas multiagentes de tempo real.**

Discordo totalmente	Discordo	Neutro	Concordo	Concordo totalmente
☐	☐	☐	☐	☐

5. **Quais dificuldades você encontrou em utilizar o metamodelo?**

6. **Você teria alguma sugestão de melhoria para o metamodelo? Qual(is)?**

B.3 Problema prático

Atividade Prática do Metamodelo - Problema

Você foi escalado para representar parte dos requisitos de um time feito em um sistema multiagente em tempo real (SMART) que irá participar da primeira Copa dos Agentes em Tempo Real. Os times são compostos por 3 agentes, que jogam em um campo quadrado durante dois tempos de 5 minutos com intervalo de 1 minuto para a troca de lado/estratégia dos times.

No SMART deve haver um papel de agente de tempo real (ART) responsável por exercer a função de goleiro. O principal objetivo do agente que assumir esse papel é defender o gol. Além disso, o mesmo deve estar o tempo todo monitorando a bola e percebendo a localização da mesma a cada milésimo de segundo. Quando houver a percepção de uma jogada ofensiva do time adversário que possa levar a um gol, o goleiro deve analisar e calcular em tempo real a trajetória da bola e realizar ações com a intenção de evitar que a bola entre no gol. Essas ações devem obedecer a um intervalo de tempo determinado, não podendo exceder 5 milésimos de segundos. O agente deve se movimentar nem tão lentamente que não se posicione a tempo de defender o gol e nem tão depressa que o adversário perceba sua posição defensiva e mude o ângulo do ataque.

Também é responsabilidade do goleiro realizar o passe da bola a partir:

- Saída da bola pela linha de fundo, em casos em que o último toque de bola ser feita pelo adversário.
- Em uma defesa em que o goleiro realize a mesma e permaneça com a bola;

Além disso, deve haver um papel responsável por interpretar a posição de atacante. O principal objetivo desse papel é marcar gols no time adversário. Também é necessário que nesse papel, ao perceber que está com o domínio da bola, o agente analise e tome decisões em tempo real para fazer um gol, em que as decisões devem ser feitas de forma não tão lenta, sendo no máximo 4 milésimos de segundos. As ações temporais desse papel são de chutar (ou não) a bola para o gol ou passar a bola para seu aliado. Essas decisões devem ser realizadas em tempo hábil.

Também deve haver um papel em tempo real interpretando a posição de zagueiro. O seu principal objetivo é marcar os atacantes, evitando que estes criem possibilidades de gol. Para isso é necessário monitorar o tempo da bola e planejar estratégias em tempo real para se adiantar (marcando o jogador) ou/e tentar tomar a bola do adversário antes que o adversário seja mais rápido.

Também deve ser modelado o papel de juiz. O principal objetivo desse papel é monitorar o andamento da partida, além de perceber e agir em tempo real nas seguintes situações:

- Sinalizar o final do primeiro tempo, o início e o final do segundo tempo (5 min cada);
- Reiniciar a partida em menos de 10 segundos após o gol;
- Sinalizar o gol de algum dos times e meio de campo a partir de algum gol;
- Marcar as faltas e as saídas de bola;

APÊNDICE C – APRESENTAÇÃO UTILIZADA NO GRUPO FOCAL

Extensão de um metamodelo para a representação de requisitos com restrições temporais

Pampa Federal University (UNIPAMPA)
Intelligent Software Engineering Laboratory

November 17, 2019

Abertura

Obrigado por Participar!

O que é um grupo focal?

Um grupo focal é um método de pesquisa que reúne pessoas em uma sala para prover feedback sobre algo.

Como funciona um grupo focal?

Objetivo

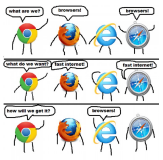
Aplicar, discutir e avaliar a extensão do metamodelo de [Goedes, 2012] para a representação de requisitos expostos a restrições temporais.

Agentes em tempo real:

- Para [Qasim, 2017] são agentes cujas **responsabilidades** podem ser limitadas por restrições de tempo.
- Já para [Ashamalla, 2017] são agentes que **consideram** o tempo para atingir seus objetivos.
- Para [Carrascosa et al., 2005] as restrições dos ARTs podem ser "**hard**" e/ou "**soft**".

Agentes em tempo real:

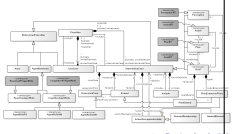
- [DiPippo et al., 2001] um ATR ao tentar atingir seus objetivos pode possivelmente impactar a **qualidade** dos seus resultados.
- Para [Bajo et al., 2008] um ATR deve **garantir** o cumprimento de suas restrições de tempo e, ao mesmo tempo, deve tentar **atingir** suas metas ou objetivos.
- Um ATR que **não cumpre** seus objetivos perde a sua **usabilidade** [Qasim et al., 2015].
- [Dragoni et al., 2018] é um processo em execução cujo sucesso depende não apenas da **totalidade** de seu objetivo, mas também do **momento** em que suas ações são **executadas**.



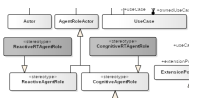
Onde são aplicados?

- Agentes físicos;
  - Robôs;
  - Carros autônomos;
- Sistemas críticos;
  - Bolsa de valores;
  - Sistemas distribuídos;
  - Troca de mensagens;
  - Controle de desastres;

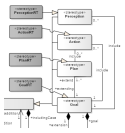
Metamodelo



Metamodelo



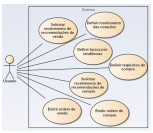
Metamodelo



Exemplo

Um ambiente em que vários agentes em tempo real são responsáveis por coordenar e realizar recomendações inteligentes relacionadas a compra e venda de ações na bolsa de valores.
Em linhas gerais, no ambiente se encontram quatro tipos diferentes de papeis de agentes: um **AgenteUsuario**, um **AgenteCotacao**, um **AgenteTendencia** e um **AgenteCompraEVenda**, onde as restrições de tempo não podem passar de 1 milissegundo.

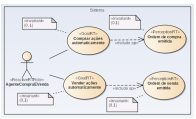
Exemplo



Exemplo

O papel **AgenteCompraEVenda** é responsável pelas **compras e vendas de ações**, em que esse agente pode agir de forma autônoma se o usuário humano tiver expressado ao **AgenteUsuario** que as transações podem ser feitas automaticamente.

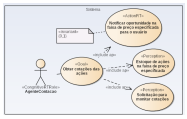
Exemplo



Exemplo

O papel **AgenteCotacao** tem a capacidade de obter cotações de ações em um setor específico do mercado. Ele também pode monitorar um estoque específico para uma faixa de preço específica.

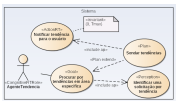
Exemplo



Exemplo

O papel **AgenteTendencia** procura tendências específicas do mercado, em que cada **AgenteTendencia** pode ser responsável por um determinado tipo de tendência, como por exemplo um aumento nos estoques do setor de biotecnologia.

Exemplo



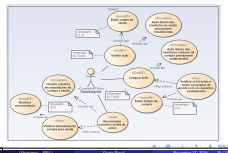
Exemplo

O papel **AgenteUsuario** se comunica com o usuário humano para determinar seus requisitos, como a quantidade de dinheiro a ser gasto e as preferências do setor de mercado.

- Fazer a recomendação ao usuário;
- Notificar o *AgenteCompraEVenda* para realizar a transação.

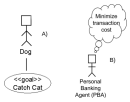
O **AgenteUsuario** também se comunica com os outros agentes no sistema para poder fazer recomendações de compra e venda ao Usuário.

Exemplo



Instrumento 01

Responder o questionário



Análise SWOT

Contribuições Finais

Fiquem à vontade para contribuir.

Obrigado por Participar!

References

Ashamalla, A. N. N. (2017).
A requirement modelling framework for real-time multi-agent systems.
PhD thesis.

Bajo, J., Julián, V., Corchado, J. M., Carrascosa, C., de Paz, Y., Botti, V., and de Paz, J. F. (2008).
An execution time planner for the artis agent architecture.
Engineering Applications of Artificial Intelligence, 21(5):769–784.

Carrascosa, C., Terrasa, A., Fabregat-Pinilla, J., and Botti, V. J. (2005).
Behaviour management for real time agents.
Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial, 9(26):39–48.

DiPippo, L. C., Fay-Wolfe, V., Nair, L., Hodys, E., and Uvarov, O. (2001).
A real-time multi-agent system architecture for e-commerce.

ÍNDICE