**UNIVERSIDADE FEDERAL DO CEARÁ - CAMPUS DE QUIXADÁ**
**CURSO DE SISTEMAS DE INFORMAÇÃO**

# PSMAR: Processo de Desenvolvimento de Sistemas Multiagentes com Agentes Racionais

## Projeto de Pesquisa

**Rendley Arnou Xavier**

**Orientador:**
**Prof°. Dr. Enyo José Tavares Gonçalves**

**QUIXADÁ**

**Outubro, 2020**

# SUMÁRIO

# 1 INTRODUÇÃO

Um agente é uma entidade de software capaz de perceber seu ambiente através de sensores e atuar nesse ambiente através de efetuadores (RUSSELL e NORVIG, 1995). Os agentes são sistemas de computador com duas importantes capacidades. Primeiro, eles são pelo menos até certo ponto capazes de *ação autônoma* - de decidir por si mesmos o que precisam fazer para satisfazer seus objetivos de projeto. Eles são capazes de interagir com outros agentes não simplesmente por troca de dados, mas envolvendo-se de forma análoga ao tipo de atividade social que nos utilizamos em nosso cotidiano: cooperação, coordenação e negociação (WOOLDRIDGE, 2002). Sistemas Multiagentes (SMA) é subárea de Inteligência Artificial (IA) que investiga o comportamento de um conjunto de agentes autônomos objetivando a solução de um problema que está além da capacidade de um único agente (JENNINGS, 1996).

Há diversas arquiteturas de agentes definidas na literatura, as quais representam o tipo/classificação de uma classe de agentes. RUSSELL e NORVIG (1995) apresentam quatro arquiteturas de agentes (agente reativo simples, agente reativo baseado em conhecimento, agente baseado em objetivo e agente baseado em utilidade) conhecidas como agentes racionais. O desenvolvimento de SMA com agentes racionais vem sendo tratado por soluções propostas a nível de modelagem de requisitos e projeto, e *framework* de desenvolvimento. *iStar4RationalAgents* (GONÇALVES *et al.* 2019) é uma extensão de *iStar 2.0* (DALPIAZ *et. al*. 2016) para modelagem de SMA com agentes racionais no nível de requisitos. A linguagem de modelagem *MAS-ML 2.0* (GONÇALVES *et* al. 2015) visa dar suporte à modelagem de SMA com agentes racionais a nível de projeto. *JAMDER* (LOPES *et al.* 2018) é um *framework* baseado em *JADE* (*Java Agent Development*) (JADE, 2020) para apoiar a codificação de SMA com agentes racionais.

Por outro lado, um processo de *software* é um conjunto de atividades que leva a produção de um produto de *software*. Essas atividades envolvem o desenvolvimento de *software* propriamente dito. As quatro atividades básicas do processo - especificação, desenvolvimento, validação e evolução - são organizadas de modo diferente nos diversos processos de desenvolvimento de modo a atender suas particularidades (SOMMERVILLE, 2007).

Os trabalhos de GONÇALVES et al. (2019), GONÇALVES et al. (2015) e LOPES et al. (2018) foram propostos de forma isolada, o que torna a integração das técnicas algo não trivial. Portanto, há a necessidade de sistematizar o uso destes resultados por meio da criação

de uma abordagem que os integre e que preencha lacunas do desenvolvimento, como a especificação de requisitos de forma textual, o mapeamento entre a fase de requisitos e projeto e a definição de atividades de apoio a testes do SMA. Assim sendo, um processo de desenvolvimento pode contribuir neste contexto. Este trabalho propõe o PSMAR, um Processo para conduzir o desenvolvimento de SMA com Agentes Racionais de modo a integrar as técnicas já existentes e propor tarefas necessárias ainda não abordadas por resultados anteriores.

## 1.1 OBJETIVOS

### 1.1.1 Objetivo Geral

Criar um processo de desenvolvimento para apoiar o desenvolvimento de sistemas multiagentes com agentes racionais.

### 1.1.2 Objetivos Específicos

I. Integrar técnicas já existentes para desenvolvimento de sistemas multiagentes com agentes racionais a nível de requisitos, projeto e implementação;
II. Propor tarefas/artefatos para as etapas do ciclo de desenvolvimento ainda não abordadas em trabalhos anteriores;
III. Modelar e disponibilizar o processo utilizando notação adequada;
IV. Avaliar o processo criado com especialistas.

# 2 TRABALHOS RELACIONADOS

Técnicas e ferramentas de engenharia de *software* são necessárias ao longo do ciclo de vida do *software*, e se fazem relevantes para conduzir com êxito o processo de desenvolvimento de *softwares*. A engenharia de *software* orientada a agentes tem como objetivo prover técnicas e ferramentas para o desenvolvimento de SMA (CASTRO *et al.*, 2006). Os trabalhos a seguir são abordagens que apoiam o desenvolvimento de SMA.

*MAS-CommonKADS* (IGLESIAS, C. A.; GARIJO, 2005) é uma metodologia da engenharia do conhecimento amplamente utilizada que estende *CommonKADS* (SCHREIBER et al., 2000). Segundo (SCHREIBER et al., 2000), *MAS-CommonKADS* é composta de cinco fases, que vão desde a contextualização do sistema até o desenvolvimento e manutenção. As fases do processo são: Contextualização, Análise, Projeto,

Desenvolvimento e Teste, Operação e Manutenção. *MAS-CommonKADS* possui sete modelos que buscam descrever as características de um agente e seus comportamentos sociais no *MAS*: modelo de agente, modelo de tarefas, modelo de coordenação, modelo de comunicação, modelo de experiência, modelo de organização e modelo de projeto. *MAS-CommonKADS+* (MORAIS II, 2010) é uma extensão que acrescenta a representação de agentes racionais a nível de projeto ao *MAS-CommonKADS*.

*MESSAGE* (HENDERSON-SELLERS e GIORGINI, 2005) segue a abordagem iterativa e incremental do *RUP*, que envolve aprimoramento progressivo dos requisitos, plano, design e implementação. Em seu processo de desenvolvimento o *MESSAGE* consiste nas fases de análise e projeto. O objetivo da fase de análise é gerar uma especificação do sistema que descreve o problema a ser solucionado, de modo a gerar um modelo abstrato que auxilia na validação e no desenvolvimento do projeto (HENDERSON-SELLERS e GIORGINI, 2005). Essa fase gera cinco modelos: modelo de organização, modelo de agentes/papéis, modelo de metas/tarefas, modelo de domínio, modelo de interação. A fase de projeto transforma os modelos gerados na análise em entidades que podem ser codificadas em uma plataforma de agentes. Esta fase é dividida em projeto de alto nível e projeto detalhado. O projeto de alto nível especifica os agentes do sistema e quais papéis são de sua responsabilidade e o projeto detalhado busca mapear os modelos de alto nível para conceitos computacionais específicos de plataformas.

*Prometheus* (PADGHAM; WINIKOFF, 2005), é uma metodologia de desenvolvimento que fornece mecanismos para a análise e projeto de *SMA* baseados em arquiteturas *BDI*. O processo de desenvolvimento é dividido em três fases: especificação do sistema, projeto arquitetural e projeto detalhado. A fase de especificação do sistema tem como finalidade definir os requisitos do sistema por meio da descrição das metas do sistema, dos cenários, das funcionalidades e do ambiente, que é descrito através de suas percepções, ações e dados externos. O projeto arquitetural é onde os artefatos gerados na fase anterior são modelados por meio dos diagramas de acoplamento e de conhecimento, diagramas de interação e do diagrama geral do sistema. A fase de projeto detalhado tem como objetivo descrever o projeto interno do agente através de suas capacidades, planos, eventos e dados. Uma capacidade pode conter planos, eventos e dados.

A tabela 1 faz uma comparação entre os processos das metodologias abordadas nos trabalhos relacionados com o trabalho proposto. Na linha (cabeçalho) são listadas as abordagens: MAS-CommonKADS+, MESSAGE, Prometheus, PSMAR.

Na segunda linha, é analisado se a respectiva abordagem foi modelada utilizando linguagem de modelagem de processo, sendo que apenas o PSMAR cumpre com este item. Na terceira linha, é verificada a cobertura da tarefa de especificação textual de requisitos, sendo que somente o PSMAR cumpre com este item da análise.

Na quarta linha, é analisado se cada abordagem define a modelagem de requisitos, sendo identificado que todas as abordagens contemplam. Na quinta linha, é verificado que a modelagem de projeto é tratada em todas as abordagens analisadas. Na sexta linha, é destacado que a geração de código é utilizada somente pelo PSMAR dentre as abordagens analisadas.

Na sétima linha da Tabela 1 é analisada a presença da fase de teste do SMA no qual apenas as metodologias *MESSAGE e PSMAR* abrangem esta fase. Por fim, a oitava e última linha analisa se cada uma das abordagens trata agentes racionais, sendo que apenas *MAS-CommonKADS+* e *PSMAR* foram assinaladas. Vale destacar que MAS-CommonKADS+ aplicou os conceitos de agentes racionais somente a nível de modelagem de requisitos e de projeto.

| **Análise/Abordagem** | **MAS-CommonKADS+** | **MESSAGE** | **Prometheus** | **PSMAR** |
|---|---|---|---|---|
| **Modelada utilizando linguagem de modelagem de processo** | | | | X |
| **Especificação textual de requisitos** | X | | X | X |
| **Modelagem de requisitos** | X | X | X | X |
| **Modelagem de projeto** | X | X | X | X |
| **Geração de código** | | | | X |
| **Atividade de testes** | | X | | X |
| **Agentes racionais** | X | | | X |

Tabela 1: Comparação entre os processos dos trabalhos relacionados com o trabalho proposto. Fonte: Elaborada pelo autor.

# 3 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão apresentados os conceitos utilizados para a realização deste trabalho. A Seção 3.1 descreve conceitos relacionados a SMA e agentes racionais e a Seção 3.2 apresenta alguns trabalhos que apoiam o desenvolvimento de SMA com agentes racionais

Um agente é um sistema computacional situado em um ambiente e que é capaz de efetuar ações autônomas neste ambiente a fim de alcançar seus objetivos (WOOLDRIDGE, 2007). Agente racional é definido como uma entidade de software autônoma que percebe seu ambiente de atuação por meio de sensores, processa essas informações e conhecimentos, e atua no ambiente por meio de atuadores visando realizar algum objetivo, conforme estabelecido em uma medida de avaliação de desempenho especificada pelo projetista do agente (RUSSEL e NORVIG, 1995). A Figura 3.1 ilustra esta definição.

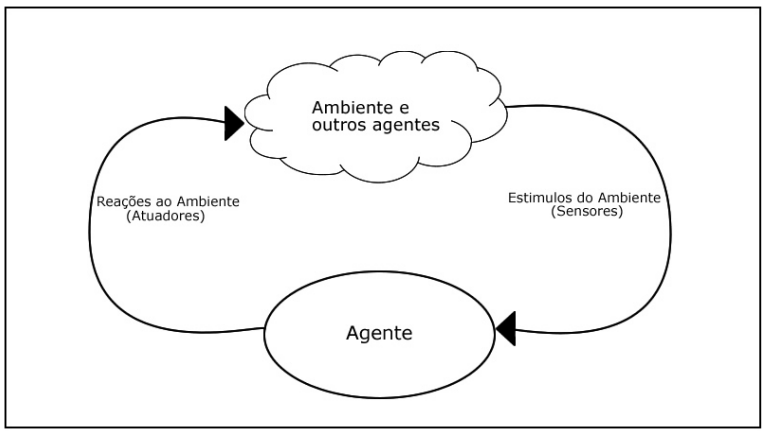



Figura 3.1: Visão geral de um agente. (MORAIS II, 2010)

WOOLDRIDGE (2002) define algumas propriedades a serem observadas em um agente, são elas:

- **Autonomia** – os agentes devem executar a maior parte de suas ações sem interferência direta de agentes humanos ou de outros agentes computacionais, possuindo controle total sobre suas ações e estado interno;
- **Habilidade social** – os agentes devem poder interagir com outros agentes (humanos ou computacionais), para completarem a resolução de seus problemas, ou ainda para auxiliarem outros agentes;
- **Reatividade** – os agentes devem perceber e reagir a alterações nos ambientes em que estiverem inseridos;
- **Pró-atividade** – os agentes, além de atuarem em resposta às alterações ocorridas em seu ambiente, devem apresentar um comportamento orientado a objetivos, tomando iniciativas quando considerarem apropriado;

- **Adaptação** – os agentes devem poder mudar o seu comportamento devido a uma experiência anterior;
- **Mobilidade** – os agentes devem poder se movimentar de uma máquina para outra.
- **Persistência ou continuidade temporal** – capacidade do agente de manter um estado interno conciso através do tempo, isto é, o agente estar continuamente executando um processo.

Alguns dos motivos para utilização de SMA apresentados por BITTENCOURT (1998) são:

- Aumentar a eficiência e a velocidade do sistema;
- Melhorar a adaptabilidade, a confiabilidade e a autonomia do sistema;
- Permitir a integração de sistemas inteligentes existentes de maneira a aumentar a capacidade de processamento e, principalmente, a eficiência na solução de problemas.
- Reduzir os custos de desenvolvimento e manutenção;

RUSSELL E NORVIG (1995) descrevem quatro arquiteturas de agentes que levam em consideração não apenas como realizar suas ações, mas também as informações de como o agente irá evoluir e como aquelas ações irão modificar o ambiente, definindo assim um conceito mais racional do agente, são elas:

- Agentes reativos simples: são os agentes que percebem o ambiente e agem baseados nas suas percepções atuais sem considerar o histórico de suas percepções. Geralmente são providos com uma base de conhecimento formada por regras Se-Então, sendo que seu comportamento está totalmente codificado nessas regras;
- Agentes reativos baseados em modelos: são agentes que guardam estados do ambiente e que sabem como o ambiente evolui em função do tempo e em função de suas ações. Portanto as regras de produção podem se basear tanto na sequência de percepção quanto no estado do ambiente para decidir o que agente deve fazer;
- Agentes baseados em objetivos: este tipo de agente toma suas decisões sempre levando em consideração a tentativa de alcançar seus objetivos;
- Agentes baseados em utilidade: pode acontecer de existirem várias sequências distintas de ações que levem o agente a atingir seu objetivo. Então esse tipo

de agente toma suas decisões baseado na função de utilidade que melhor se adeque para a resolução do seu objetivo;

## 3.1 Apoio ao desenvolvimento de agentes racionais

Nesta seção serão apresentados alguns trabalhos que apoiam o desenvolvimento de SMA com agentes racionais. Na subseção 3.2.1 é detalhada a extensão *iStar4RationalAgents*. Na subseção 3.2.2, *MAS-ML 2.0* é apresentada. *JAMDER*, um framework baseado em *JADE* para desenvolvimento de SMA com agentes racionais, é apresentado na subseção 3.2.3.

### 3.1.1 iStar4RationalAgents

O *iStar4RationalAgents* é uma extensão proposta à linguagem de modelagem *iStar*. O *iStar* é uma linguagem de modelagem baseada em objetivos, usada para modelar software no nível de requisitos (GONÇALVES *et al*, 2019).

Na estrutura do *iStar*, as partes interessadas são representadas como atores que dependem uma da outra para atingir seus objetivos, executar tarefas e fornecer recursos. Cada objetivo é analisado do ponto de vista do ator, resultando em um conjunto de dependências entre pares de atores. Os elementos são classificados como Elementos intencionais (objetivo, tarefa e recurso), atores (ator geral, função, cargo e agente) e *links* (*links* de meios finais, decomposição, contribuição e ator). Esses elementos são representados em dois modelos: Dependência Estratégica (*SD*) e Razão Estratégica (*SR*). O modelo *SD* descreve os links e dependências externas entre os atores organizacionais. O modelo *SR* permite uma análise de como os objetivos podem ser alcançados através de contribuições dos diversos atores (GONÇALVES *et al.*, 2019).

O modelo de *SD* deve ser criado para representar os agentes, papéis, organizações e ambientes envolvidos no *SMA* e representar o relacionamento entre eles. Um agente pode desempenhar um papel, habitar um ambiente e fazer parte de uma organização. As dependências entre agentes, funções, organizações e ambientes também podem ser expressas. O principal objetivo desta modelagem é representar o *SMA* a ser desenvolvido na fase inicial, quando o *SMA* começar a ser proposto e as decisões sobre os elementos intencionais dos agentes, como objetivo, tarefa, planejamento, percepção e detalhes internos das intenções dos agentes elementos ainda não estão claros (GONÇALVES *et al.*, 2019).

Além disso, o modelo *SR* deve ser criado para representar os detalhes internos dos agentes, papéis, organizações e ambientes envolvidos no *SMA* e representar o relacionamento

entre eles. O modelo *SD* é o ponto de partida para a criação deste modelo (GONÇALVES *et al.*, 2019).

A extensão foi aplicada na ferramenta de modelagem *piStar*. A ferramenta estendida foi testada pela modelagem da ilustração, não foram encontradas correções a serem feitas. A ferramenta está disponível em www.cin.ufpe.br/~ler/piStar4rationalagents. A Figura 3.2.1.1 mostra uma visão geral do piStar4rationalagents (GONÇALVES *et al.*, 2019).

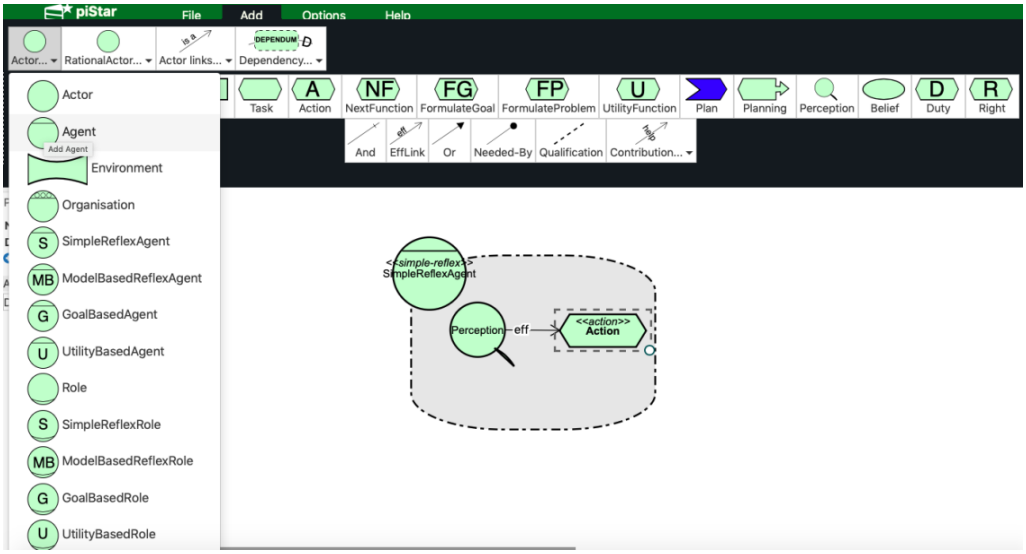



Figura 3.2.1.1 Visão geral do *piStar4rationalagents*. (GONÇALVES *et al.*, 2019)

Na ilustração foi modelado o *SMA* para *MOODLE* com a extensão proposta. O *MOODLE* (ambiente modular de aprendizagem dinâmica orientada a objetos) é amplamente utilizado em cursos a distância. Este software está disponível em uma distribuição padrão que pode ser personalizada. Esse ambiente é utilizado em cursos da Universidade Aberta do Brasil (UAB), uma universidade do governo brasileiro que oferece cursos em educação a distância. Assim, foi modelado um *SMA* para os cursos da UAB oferecidos em parceria com a Universidade Estadual do Ceará (UAB / UECE) (GONÇALVES *et al.*, 2019).

Cinco agentes foram considerados na modelagem:

- **Agente Auxiliar:** Este agente é um tipo simples de reflexivo e possui uma lista de vários insights sobre as dificuldades que o usuário possui e, antes disso, escolhe a ação apropriada. Esse agente percebe em que momento o usuário está e, ao mesmo tempo, oferece dicas sobre como fazer o melhor uso de uma funcionalidade específica, especificamente, a ação para um trabalho específico;
- **Agente de Aprendizado Associado:** Esse tipo de agente deve poder escolher independentemente entre um intervalo predeterminado de estratégias de

interação afetiva, como mensagens de suporte. Apresenta mensagens encorajadoras (reforço positivo) quando o usuário, por meio das interações manifestas, fornece evidências fáceis de seguir a discussão e / ou as tarefas e / ou conteúdos propostos, e mesmo quando o aluno apresenta notas e iterações acima da média em sua classe ou grupo de trabalho. Devido à necessidade de manter anotações de aula para comparação e enviar mensagens rapidamente, esse agente é caracterizado como um agente reflexivo baseado em modelo;

- **Agente Pedagógico:** Esse agente deve poder acompanhar o aluno nas diferentes disciplinas que participam para contribuir com o usuário através de dicas, sugestões e mensagens relacionadas ao tópico em andamento e não apenas à natureza afetiva das mensagens (suporte). É um agente baseado em objetivos, porque precisa criar uma estratégia de estudo, sugerindo disciplinas para o aluno com base nas disciplinas que o aluno está realizando. Este agente é modelado como um agente baseado em objetivos;
- **Agente do grupo:** este agente é um agente baseado em utilidade. Deverá ser capaz de ajudar autonomamente usuários, estudantes e educadores, na composição de grupos de trabalho, levando em consideração temas de afinidade ou perfis de aprendizagem. Para isso, deve considerar certos critérios estabelecidos por um treinador de uma ou mais classes ou pelo usuário interessado em integrar os grupos de trabalho;
- **Agente de Pesquisa:** Este agente encontra material extra (páginas, projetos e outros objetos digitais) relacionados aos cursos no *MOODLE* e enviado aos alunos. Este agente é modelado como um agente padrão do *MAS-ML*.

Inicialmente, o engenheiro de requisitos deve modelar o diagrama estendido para *SD* com a representação do tipo de agentes com o estereótipo específico. Seus papéis devem ser representados e também a organização e o ambiente. Por fim, o link de participação deve ser usado para representar os papéis desempenhados pelos agentes e o ambiente em que eles habitam. A participação também é usada para representar a propriedade dos agentes por suas organizações. O diagrama *SD* do SMA do *MOODLE* representa o *MOODLE* como o ambiente em que os agentes podem perceber e agir. Foi modelada uma organização para coordenar os comportamentos dos agentes. Finalmente, o diagrama representa quatro agentes e duas funções desempenhadas por dois deles. A Figura 3.2.1.2 mostra o diagrama *SD* para o SMA do *MOODLE* (GONÇALVES *et al.*, 2019).

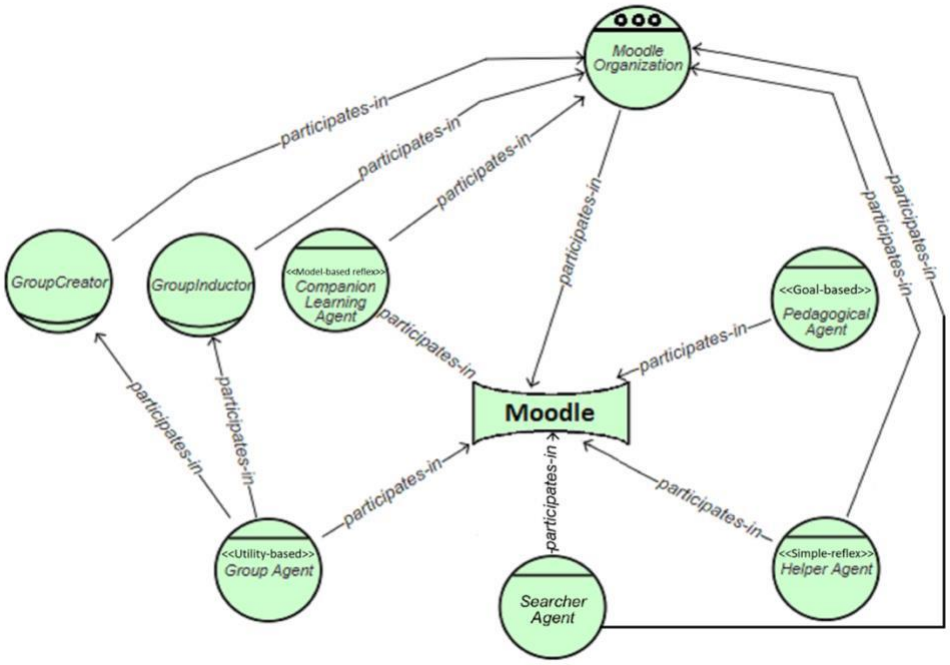



Figura 3.2.1.2 Modelagem do modelo *SD* do *iStar* para *MOODLE* usando a nova extensão *iStar*. (GONÇALVES *et al.*, 2019).

O diagrama *SR* mostra os elementos internos dos agentes. O conjunto de elementos relacionados depende do tipo de agente. As percepções dos agentes estão relacionadas ao meio ambiente por um relacionamento de dependência. A Figura 3.2.1.3 mostra o diagrama *SR* para o SMA do *MOODLE* (GONÇALVES *et al.*, 2019).

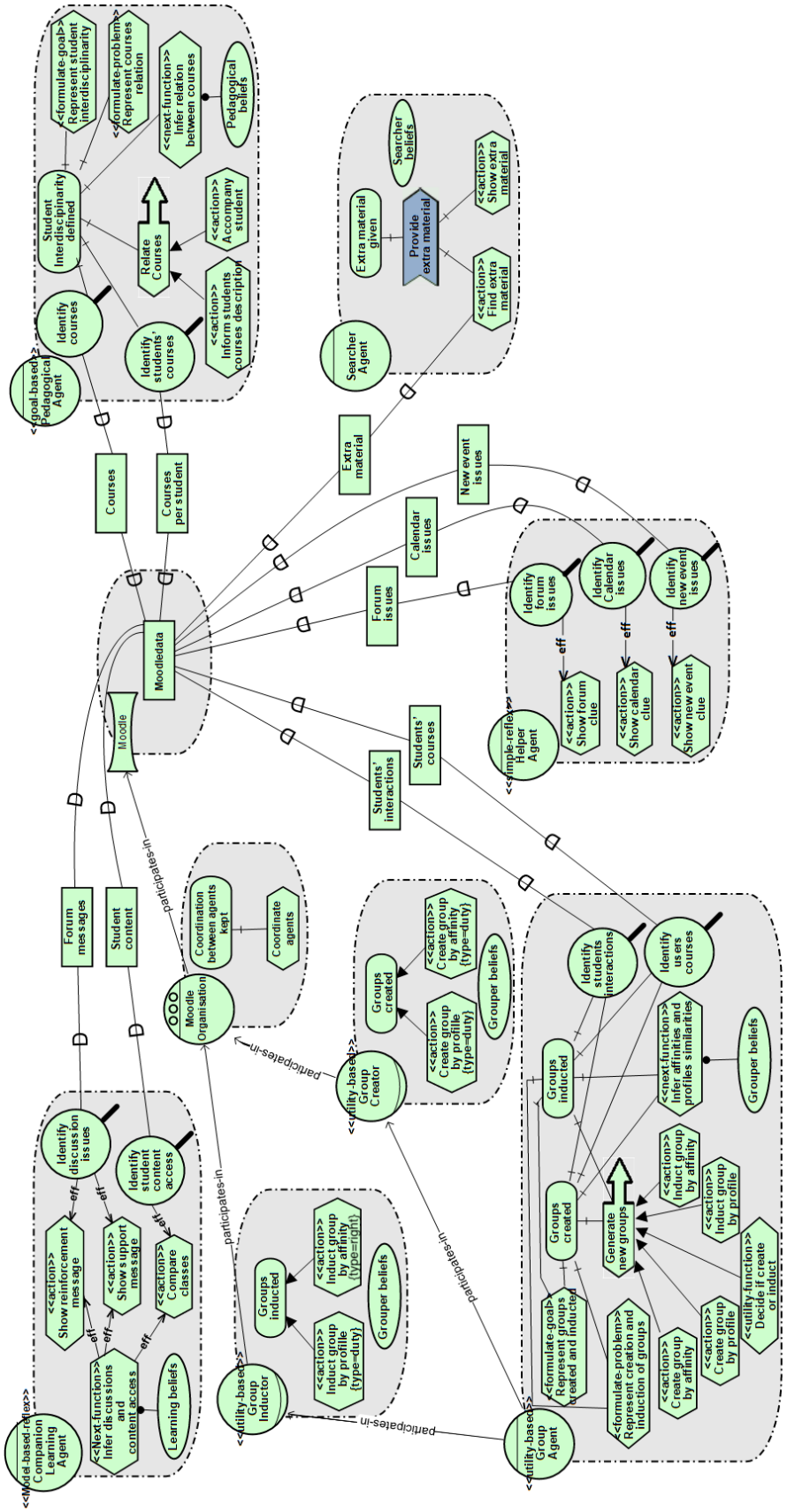



Figura 3.2.1.3 Modelagem do modelo *iStar SR* para *MOODLE* usando a nova extensão *iStar*. (GONÇALVES *et al.*, 2019).

### 3.1.2 MAS-ML 2.0

*MAS-ML* é uma linguagem de modelagem que implementa os conceitos de *TAO* (*Taming Agents and Objects*) e foi originalmente projetada para suportar a modelagem de agentes proativos baseados em objetivos, guiados por planos pré-estabelecidos (SILVA, 2004). *MAS-ML* modela todos os aspectos estruturais e dinâmicos definidos no metamodelo *TAO* estendendo o metamodelo *UML* (*Unified Modeling Language*). MAS-ML modela cinco diagramas, os quais são extensões dos diagramas de UML. Os diagramas estruturais definidos por *MAS-ML* são: Diagrama de Classes, Diagrama de Organização e Diagrama de Papéis. Os diagramas dinâmicos são Diagrama de Sequência e Diagrama de Atividades. Em MAS-ML agentes são modelados com objetivos, crenças, planos e ações.

MAS-ML 2.0 (GONÇALVES, 2009) é uma extensão de MAS-ML para incluir elementos internos de agentes racionais. A extensão é utilizada para apoiar a modelagem de agentes usando diferentes arquiteturas internas: Reativa Simples, Reativa Baseado em Modelo, Baseada em Objetivo e Baseada em Utilidade.

Segundo UML (2009), valores marcados, estereótipos e restrições são mecanismos de extensão. Além disso, a adaptação das metaclasses existentes e a definição de novas metaclasses também podem ser usadas. Os estereótipos e a definição de novas metaclasses foram usados para representar agentes reflexos simples, agentes reflexivos baseados em modelos, agentes baseados em objetivos com agentes planejados e baseados em serviços públicos. Seguindo as definições de arquitetura apresentadas na pesquisa, sentiram a necessidade de definir as seguintes características para a extensão: percepção, função seguinte, função formular meta, função formular problema, função formular problema, planejamento e função utilidade (Gonçalves *et al.*, 2015).

A Figura 3.2.2.1 ilustra o metamodelo MAS-ML 2.0. As novas metaclasses são representadas por meio usando retângulos brancos de linha dupla, já os novos estereótipos são representados com um retângulo branco com bordas arredondadas (GONÇALVES *et al.*, 2015).

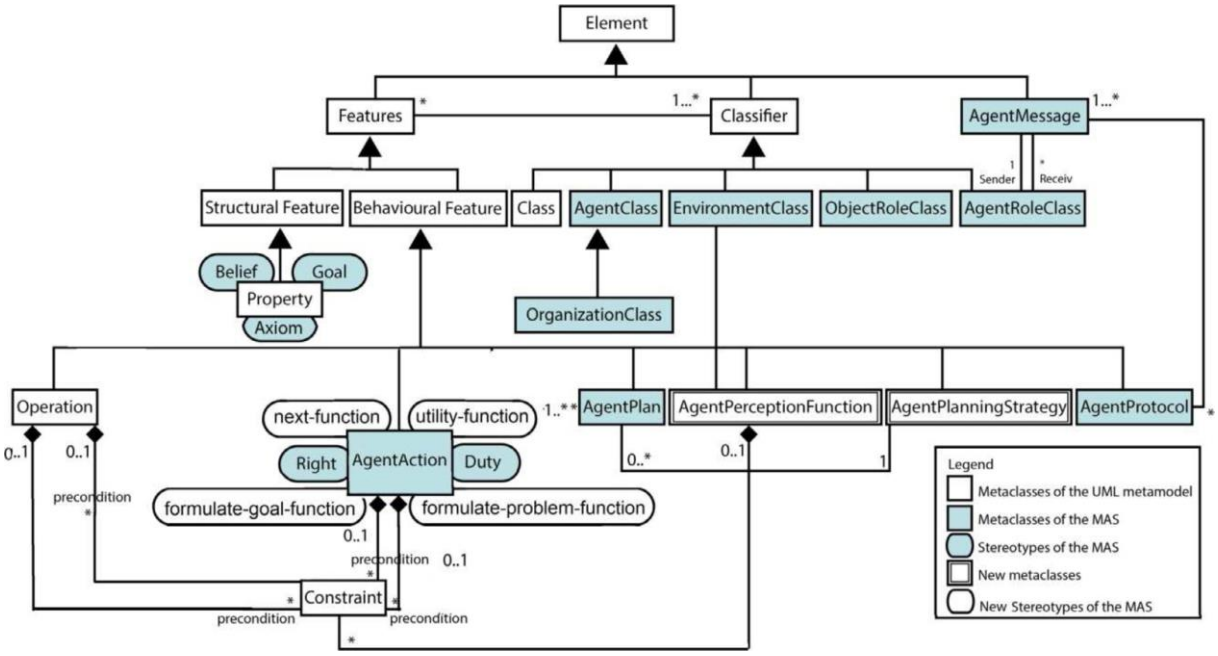



Figura 3.2.2.1 - Metamodelo da extensão do MAS-ML

Os novos recursos estruturais e comportamentais introduzidos no metamodelo MAS-ML são usados para modelar os diferentes tipos de agentes e impactam a representação da metaclasse AgentClass em diagramas estáticos (Gonçalves et al., 2015). Semelhante aos diagramas estáticos, as novas definições de agente também impactam os diagramas de sequência e de atividades.

Na Figura 3.2.2.2 são representadas as modelagens dos agentes nos diagramas estáticos. Na Figura 3.2.2.2 (I) o Agente Reativo Simples não inclui nenhum elemento estrutural no compartimento intermediário, mas no compartimento inferior são representadas as percepções e ações, orientadas por regras de condição ação e não por um plano específico. Na Figura 3.2.2.2 (II) o Agente Reativo baseado em conhecimento é representado. Assim, há a representação de crenças representando o estado, além de percepções, ações e da função próximo. Na Figura 3.2.2.2 (III) o Agente Baseado em Objetivo é apresentado, o qual possui objetivos e crenças no compartimento intermediário, e percepções, ações, função próximo, função de formulação de objetivo, função de formulação de problema e planejamento no compartimento inferior.

Na Figura 3.2.2.2 (IV) é representado o Agente Baseado em Utilidade, que consiste em uma especialização do Agente Baseado em Objetivo com o planejamento. Durante o planejamento, os agentes podem ser vinculados para atingir mais de uma meta. Nesse caso, é possível a ocorrência de objetivos conflitantes ou a existência de vários estados que atendem aos objetivos. Portanto, a função de utilidade é incorporada à estrutura do agente para avaliar o grau de utilidade dos objetivos associados. Assim, o elemento <<utility-

function>> é adicionado para representar a função utilidade responsável pela otimização do desempenho do agente.

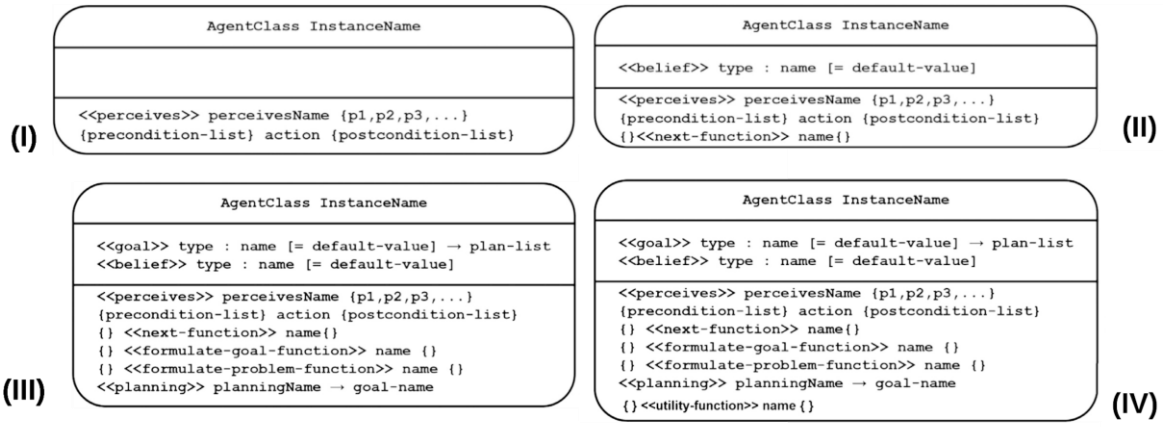



Figura 3.2.2.2 - Representação de Agentes com Diagramas Estáticos

Um papel de agente é representado em MAS-ML por um retângulo sólido com uma curva na parte inferior e possui três compartimentos separados por linhas horizontais. O compartimento superior contém o nome do papel de agente, o compartimento intermediário contém uma lista de objetivos e crenças associadas ao papel e, abaixo, uma lista de deveres, direitos e protocolos (SILVA, 2004).

Os agentes reativos não têm objetivos explícitos e, mais particularmente, os agentes reflexos simples não têm crenças. Assim, sua representação de papéis foi adaptada por GONÇALVES et al. (2015) e é ilustrada Figura 3.2.2.3, sendo o papel de agente reativo simples do lado esquerdo e de agente reativo baseado em conhecimento do lado direito.

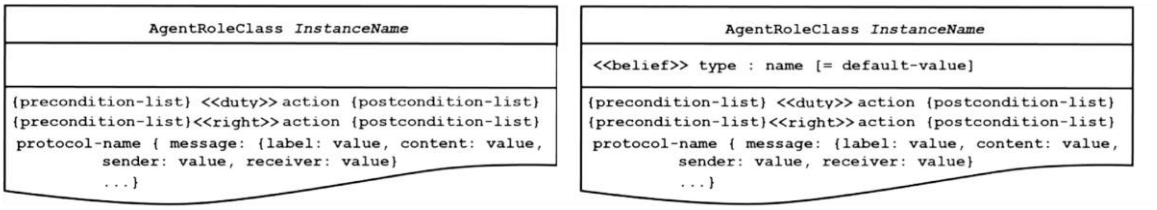



Figura 3.2.2.3 - Representação de papéis para Agente Reflexivo Simples e Agente Reflexivo Baseado em Modelo com Diagramas Estáticos

A ferramenta *MAS-ML* tool (Silva et al., 2007) (Gonçalves et al., 2015) é um *plug-in* do *Eclipse* para apoiar a modelagem da linguagem *MAS-ML 2.0* (Gonçalves et al., 2011). *MAS-ML* tool suporta a modelagem dos seguintes diagramas: diagrama de organização, diagrama de funções e diagrama de classes, com base no metamodelo *MAS-ML*. Essa ferramenta gera o arquivo masml (*XMI - XML Metadata Interchange*) que armazena a estrutura das entidades de dados e os aspectos estruturais e comportamentais definidos por *MAS-ML* 2.0.

### 3.1.3 JAMDER

*JAMDER* é uma extensão do *JADE*, um *framework* gratuito que possui uma plataforma na linguagem de programação *JAVA* e que também suporta sistemas distribuídos. No entanto, a implementação do agente no *JADE* é limitada principalmente a agentes, comportamentos e mensagens. O *JADE* também possui uma plataforma, que contém o ambiente necessário para o ciclo de vida do agente e contêineres em que os agentes residem. Esta extensão fornece a infraestrutura adequada focada na implementação de agentes de acordo com as configurações das arquiteturas internas de agentes racionais (LOPES *et al*, 2018).

Além disso, JAMDER contém uma abordagem para geração de código. O principal benefício da extensão proposta é incluir as arquiteturas de agentes racinais, entidades e relacionamentos internos do agente em uma estrutura de implementação e aumentar a produtividade por geração de código, garantindo a consistência entre o design e o código (LOPES *et al*, 2018).

A estrutura *JADE* inclui classes e serviços para facilitar a fase de codificação do *MAS*. Dentre os recursos disponíveis, alguns estão listados a seguir: serviços de publicação em páginas amarelas, serviço de localização em páginas brancas, suporte a ontologias e comunicação de protocolos compatíveis com o padrão *FIPA 6* (*Foundations of Intelligent Physical Agents*). De fato, o *JADE* é uma estrutura orientada a objetos escrita em Java (LOPES *et al*, 2018).

A arquitetura no *JADE* contém contêineres onde os agentes residem e o sistema pode ser distribuído em diferentes plataformas. Cada agente é registrado no serviço *AMS* (*Agent Management System*) fornecido pela *JADE* que garante a unicidade dos agentes. Para encontrar outros agentes, outro serviço é fornecido, o *DF* (*Directory Facilitator*) e funciona como um serviço de Páginas Amarelas (LOPES *et al*, 2018).

O ciclo de vida do *JADE* para um agente é definido de acordo com o *FIPA*. A primeira etapa é a execução do construtor do agente, seguida pela atribuição de um identificador para o agente a ser inserido no sistema. O método setup () é executado a partir do momento em que o agente inicia suas atividades. Um comportamento do agente é projetado pela substituição da configuração (LOPES *et al*, 2018).

A arquitetura orientada a modelo (*MDA*) é apresentada como uma abordagem apropriada para auxiliar na geração de código, porque o código pode ser gerado várias vezes sem comprometer o modelo. O *OMG* 7 define alguma padronização nesse processo que não

está necessariamente associada a uma plataforma específica. Assim, os conceitos podem ser aplicados a diferentes linguagens de modelagem e implementação (LOPES *et al*, 2018).

O processo de transformação ocorre através das etapas propostas pela *OMG*. Os artefatos gerados em cada uma dessas etapas são independentes de uma ferramenta específica a ser usada, por exemplo, quando o mesmo aplicativo pode ser implementado em linguagens diferentes. Os conceitos de cada uma dessas etapas são descritos a seguir (LOPES *et al*, 2018):

- *CIM* (*Computation Independent Model*): regras para requisitos funcionais;
- *PIM* (*Platform-Independent Model*): relações entre propriedades e seus relacionamentos com entidades.
- *PSM* (*Platform-Specific Model*): definição de como o sistema funcionará em uma plataforma específica.
- *PDM* (*Platform Description Model*): definição de como o *PIM* para *PSM* funcionará.

No contexto da abordagem *MDA*, é proposto o suporte automatizado à geração de código para o *SMA*. No processo proposto, são aplicados os seguintes componentes para gerar código: Ferramenta *MAS-ML tool* (*PIM*); *Framework Java* / *JAMDER* (*PSM*); e modelos *Acceleo* (*PDM*). Os arquivos de modelagem gerados pela ferramenta MAS-ML tool são a entrada para o processo de geração de código usando o *plug-in Acceleo* para suportar o conceito de *MDA*, pois permite a geração de código em diferentes linguagens de código e o desenvolvimento incremental (LOPES *et al*, 2018).

Para formalizar a geração de código no *Acceleo*, é necessário estabelecer um modelo para cada entidade por meio de uma linguagem definida pelo *OMG*, o *MTL* (*Model Transformation Language*). Quando o modelo é executado no *Eclipse*, ele precisará da representação do modelo *MAS-ML* tool (arquivos *.masml*) e da pasta de saída para armazenar as classes *JAMDER*) (LOPES *et al*, 2018).

Por exemplo, o ambiente é uma instância de uma classe que herda da classe *Environment* no *JAMDER* e sua representação ocorre através do *EnvironmentClass* na ferramenta *MAS-ML tool*. Como essa instância herda os métodos definidos em Ambiente, é necessário apenas, na geração do código, o construtor dessa classe chamar a superclasse e criar organizações, agentes e funções de agente nessa ordem. A necessidade de criar instâncias de função de agente é apenas para vincular a organização e as instâncias do agente, nas quais a construção da função de agente faz esse *link*. O mapeamento de agentes de criação e organização de corpos ocorre por meio da relação entre o ambiente e essas entidades. Os

objetos e funções de objeto também são criados no ambiente em que as funções são obtidas pela organização por meio do relacionamento de propriedade. Por sua vez, os objetos são alcançados pelo objeto através do relacionamento lúdico. A Figura 3.2.3.1 mostra o código gerado para um ambiente modelado em MAS-ML tool.

```
public class (c.name /) extends Environment {
  public (c.name /) (String name, String host, String port) {
        super(name, host, port);
   (for(i : Inhabit | c.inhabit))
       (let organ : OrganizationClass = i.org.name)
        (if (i.org -> size() > 0) )
           Organization (organ.name/) = new Organization("(organ.name/)", this, null);
           addOrganization("(organ.name/)", (organ.name/));
           (if (i.org.play -> size() > 0) )
           (let ar : AgentRoleClass = i.org.play.agentRole)
             AgentRole (ar.name/) = new AgentRole("(ar.name/)",
(ar.ownership.owner.name/), (organ.name/));
           (/let)  (/if)

           (for(ow : Ownership | i.org.ownership))
                    (for(ob : ObjectRoleClass | ow.objectRole))
                      Object (ob.play.name/) = new Object();
                      addObject("(ob.name/)", (ob.name/));
                      ObjectRole (ob.name/) = new ObjectRole("(ob.name/)",
(organ.name/), (ob.play.name/));
                    (/for)
           (/for)
            (/if)
       (/let)
    (/for)

    (for(i : Inhabit | c.inhabit))
       (if (i.agentClass -> size() > 0) )
          (let a : AgentClass = i.agentClass)
          GenericAgent (a.name /) = new (a.name /)("(a.name /)", this, null);
          AgentRole (a.play.agentRole.name/) = new
AgentRole("(a.play.agentRole.name/)", (a.play.agentRole.ownership.owner.name/),
(a.name/));
          addAgent("(a.name /)", (a.name /));
          (/let)
       (/if)
    (/for)
  }
  // Additional attributes
  (for(p : Property | c.ownedProperty))
     (p.visibility/) (p.type.toString()/) (p.name/);
  (/for)

  // Additional methods
  (for(o : Operation | c.ownedOperation))
     (o.visibility /) (o.returnValue /) (o.name /)
        ((for(p:Parameter | o.parameter) separator(',')) (p.type/) (p.name/) (/for)) ()
  (/for)
}
(/file)
(/template)
```

Figura 3.2.3.1 *Environment Template* (LOPES *et al*, 2018)

# 4 PROCEDIMENTOS METODOLÓGICOS

Neste capítulo serão descritos os passos para a execução dos artefatos referentes a este trabalho.

I. Estudo das técnicas envolvidas.
II. Definição da especificação textual de requisitos
III. Definição do mapeamento entre iStar4RationalAgents e MAS-ML 2.0
IV. Modelagem do PSMAR utilizando BPMN.
V. Avaliação do processo proposto.

O cronograma da execução dos próximos passos também é apresentado nesta seção.

## 4.1 Estudo das técnicas envolvidas

Nesta etapa foi realizado o estudo de modelos de processo de software, onde cada modelo de processo representa um processo sob determinada perspectiva e, dessa forma, viabilizar a criação do modelo proposto fundamentado nas variações desses processos e então estabelecer seus principais estágios que definem as atividades fundamentais de desenvolvimento. Além disto, os trabalhos existentes de iStar4RationalAgents, MAS-ML 2.0 e JAMDER foram estudados.

Um SMA para o MOODLE foi desenvolvido para o MOODLE utilizando estas técnicas (GONÇALVES, ARAUJO E CASTRO, 2019) e (GONÇALVES et al., 2014). Os passos seguidos neste desenvolvimento foram analisados de modo a servir como base para a modelagem do processo PSMAR. Foi identificado, além do uso das técnicas já mencionadas, o uso de uma planilha de testes utilizada durante o desenvolvimento do SMA. Vale destacar a ausência de uma especificação textual de requisitos e de um mapeamento entre iStar4RationalAgents e MAS-ML 2.0.

## 4.2 Definição da especificação textual de requisitos

Nesta etapa foram estabelecidas as atividades para sistematizar a especificação dos requisitos do *SMA* com agentes racionais. Foi criada uma abordagem de especificação textual baseada em diferentes níveis de especificação.

## 4.3 Definição do mapeamento entre iStar4RationalAgents e MAS-ML 2.0

A modelagem do SMA com agentes racionais envolve a modelagem de requisitos com iStar4RationalAgents e MAS-ML 2.0. É importante portanto mostrar um mapeamento entre as entidades destas linguagens para facilitar a conversão entre os modelos. O mapeamento entre modelos de iStar e de UML apresentado por MELO et al. (2015) foi utilizado como ponto de partida.

## 4.4 Modelagem do PSMAR utilizando *BPMN*

Nesta etapa foi feita a criação do processo de desenvolvimento de *SMA* com agentes racionais, utilizando *BPMN* uma notação da metodologia de gerenciamento de processos de negócio. Foi adotada a notação *BPMN* considerando que a mesma permite que seja ilustrado e compreendida todas as tarefas operacionais de um negócio de forma lógica, sequencial, e também é possível identificar os papéis de cada um, os eventos e todos os demais componentes de um processo.

Fazendo-se possível que profissionais de diversas nacionalidades consigam compreender a dinâmica de um fluxo operacional por meio dos símbolos utilizados pela notação.

## 4.5 Avaliação do processo proposto

Nesta etapa, o PSMAR será avaliado por especialistas em SMA por meio de um questionário online auto aplicado. As diretrizes estabelecidas por KITCHENHAM e PFLEEGER (2002) serão seguidas neste estudo.

## 4.6 Cronograma de execução

| ATIVIDADES | Nov/20 | Dez/20 | Jan/21 | Fev/21 | Mar/21 | Abr/21 |
|---|---|---|---|---|---|---|
| Correções do processo PSMAR | X | X | | | | |
| Planejamento do survey de avaliação do PSMAR | | | X | | | |
| Piloto do survey de avaliação do PSMAR | | | X | | | |
| Aplicação do survey de avaliação do PSMAR | | | | X | X | |
| Análise dos resultados do survey do PSMAR | | | | | X | |
| Escrita do TCC 2 | X | X | X | X | X | X |
| Defesa final | | | | | | X |

# 5 RESULTADOS PRELIMINARES

## 5.1 PSMAR

Esse processo é o foco elementar desta pesquisa; conduz metodicamente a metodologia de desenvolvimento, garantindo a integridade, consistência e, com intuito de reduzir falhas durante a criação de SMA com agentes racionais. A versão completa do processo pode ser acessada em: www.cin.ufpe.br/~ejtg/PSMAR. Os documentos (templates) utilizados neste processo podem ser vistos nos apêndices deste documento.

### 5.1.1 Fluxo Principal

O processo inicia quando surge a intenção de desenvolver SMA com agentes racionais. O processo principal é composto por quatro subprocessos. A Figura 5.1.1 apresenta uma visão geral do PSMAR em BPMN.

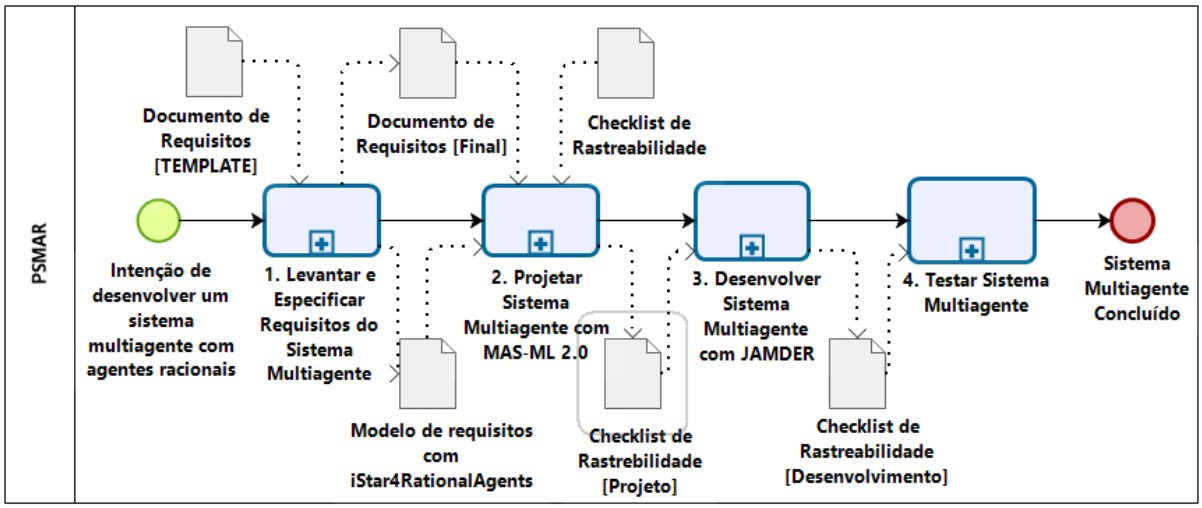



Figura 5.1.1 - Fluxo principal do PSMAR.

Os subprocessos do PSMAR encontram-se descritos a seguir:

1. **Levantar e Especificar Requisitos do SMA.** Este subprocesso consiste na identificação e descrição do SMA a nível de requisitos através do preenchimento do template de documento de requisitos e da criação do modelo de requisitos utilizando iStar4RationalAgents.
2. **Projetar Sistema Multiagente com MAS-ML 2.0.** Este subprocesso consiste na produção do modelo a nível de projeto através da modelagem do SMA com diagramas de atividade, diagrama de papéis e diagrama de organização utilizando a linguagem de modelagem MAS-ML 2.0. Esta tarefa é realizada utilizando o Eclipse com MAS-ML tool.

3. **Desenvolver Sistema Multiagente com JAMDER.** Este subprocesso consiste na implementação do SMA através do framework JAMDER. Esta tarefa é realizada utilizando o Eclipse IDE for Java Developers e os frameworks JADE e JAMDER.
4. **Testar Sistema Multiagente.** Este subprocesso consiste no teste do SMA, identificação e correção de falhas. Nesta etapa, são realizados os testes das funcionalidades de cada entidade do sistema. Caso seja identificada alguma falha, deverão ser corrigidas.

### 5.1.2 Levantar e Especificar Requisitos do SMA (Subprocesso 1)

A execução deste subprocesso inicia com a necessidade de especificar os requisitos do sistema. Ele apresenta um conjunto de tarefas para apoiar o analista de requisitos a especificar os requisitos do SMA. O subprocesso pode ser visto na Figura 5.1.2 utilizando a notação BPMN *(Business Process Modeling Notation)* para representar. O subprocesso é detalhado nos parágrafos a seguir.

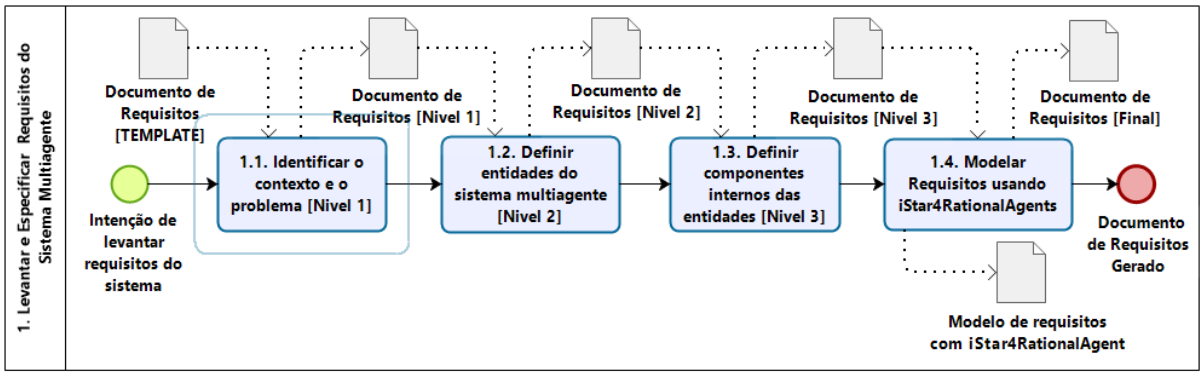



Figura 5.2.1 - Levantar e Especificar Requisitos do Sistema Multiagente.

O processo inicia a partir da intenção de levantar requisitos do SMA. A tarefa **1.1 Identificar o contexto e o problema**, recebe o template do documento de requisitos (disponível no Apêndice A) como artefato de entrada. Nesta etapa, o analista de requisitos deverá preencher o nível 1 de forma textual com intuito de descrever o sistema de forma geral.

Após finalizar a execução da tarefa 1.1 o nível 1 do documento estará preenchido, e então é iniciada a tarefa **1.2 Definir entidades do sistema multiagente** recebendo como artefato de entrada o documento com o nível 1 já preenchido. Nesta etapa, o analista de

requisitos deverá preencher o nível 2 de forma textual por meio de utilização de tabelas do documento com intuito de descrever as entidades do sistema.

Com a tarefa 1.2 finalizada o nível 2 do documento estará preenchido, por consequência, dando início a tarefa **1.3 Definir os componentes internos das entidades** recebendo como artefato de entrada o documento com o nível 2 já preenchido. Nesta etapa o analista de requisitos deverá preencher o nível 3 de forma textual por meio de utilização de tabelas com intuito de descrever os elementos intencionais das entidades do sistema, detalhando assim a arquitetura interna de cada agente.

Em seguida é iniciada a última tarefa deste subprocesso, **1.4 Modelar requisitos usando iStar4RationalAgents,** recebendo como artefato de entrada o documento com o nível 3 já preenchido. Nesta etapa o analista de requisitos deverá modelar o sistema a nível de requisitos utilizando a ferramenta piStar4rationalAgents (GONÇALVES *et al.*, 2019), disponível em www.cin.ufpe.br/~ler/piStar4rationalagents, com base nos conceitos e regras da linguagem.

Ao fim da execução deste subprocesso o analista de requisitos terá gerado como artefato de saída o documento de requisitos com os 3 níveis preenchidos juntamente com a modelagem a nível de requisitos do SMA.

### 5.1.3 Projetar Sistema Multiagente com MAS-ML 2.0 (Subprocesso 2)

Este subprocesso é iniciado após o fim da execução do subprocesso 1. Ele apresenta um conjunto de tarefas para amparar o projetista a criar o projeto do SMA. O subprocesso pode ser visto na Figura 5.1.3 e é detalhado nos parágrafos a seguir.

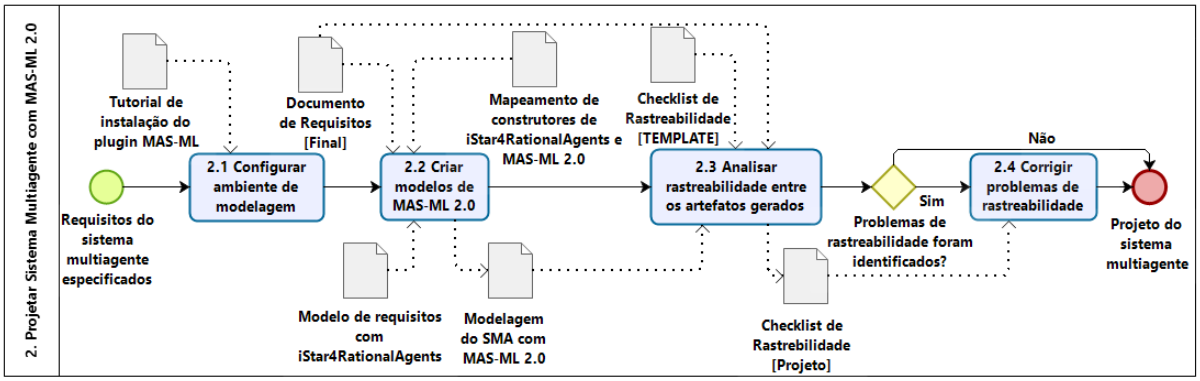



Figura 5.1.3 - Projetar Sistema Multiagente com MAS-ML 2.0.

Após o fim do subprocesso 1, inicia-se o subprocesso 2 com a tarefa 2.1 Configurar ambiente de modelagem recebendo como artefato de entrada documento PDF de tutorial de instalação do plugin MAS-ML 2.0 (disponível no Apêndice G). Em seguida na tarefa 2.2 Criar modelos de MAS-ML 2.0 recebe como artefatos de entrada o documento de requisitos final, o modelo de requisitos com iStar4RationalAgents, a planilha de mapeamento de construtores de iStar4rationalAgents e MAS-ML 2.0 (disponível no Apêndice B). Os modelos de classe, papéis, organização, sequência e atividades serão gerados tomando como base o documento de requisitos e a modelagem de iStar4RationalAgents gerados no subprocesso 1, bem como considerando a planilha de mapeamento de construtores de iStar4RationalAgents e MAS-ML 2.0.

A modelagem do SMA com MAS-ML 2.0 é gerada como artefato de saída pela tarefa 2.2 e é utilizada como entrada para a tarefa **2.3 Analisar rastreabilidade dos artefatos gerados.** A tarefa 2.3 receberá ainda um segundo artefato de entrada, trata-se do template de checklist de rastreabilidade (disponível no Apêndice C). Nesta tarefa, os artefatos gerados pelo processo de desenvolvimento serão a analisados com base no checklist de rastreabilidade.

Em seguida, é tomada uma decisão com base nas conclusões da análise de rastreabilidade. Caso problemas de rastreabilidade sejam identificados, a tarefa **2.4 Corrigir problemas de rastreabilidade** é executada, recebendo o checklist de rastreabilidade como entrada. Se não, o subprocesso é finalizado com o projeto do SMA gerado.

### 5.1.4 Desenvolver Sistema Multiagente com JAMDER (Subprocesso 3)

Este subprocesso é iniciado após o fim da execução do subprocesso 2. Ele apresenta um conjunto de tarefas para amparar o programador a implementar o SMA. O subprocesso pode ser visto na Figura 5.1.4. O subprocesso é detalhado nos parágrafos a seguir.

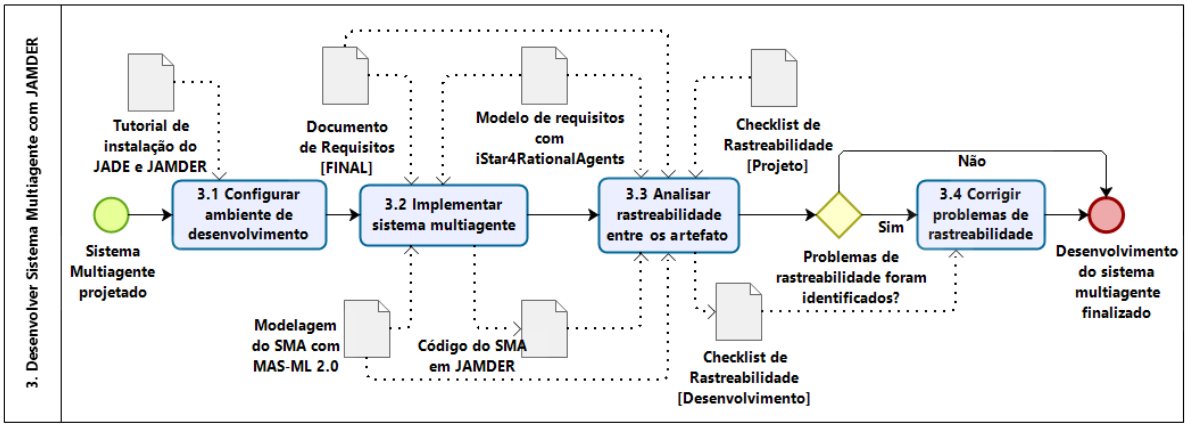



Figura 5.1.4 - Desenvolver o Sistema Multiagente com JAMDER.

O subprocesso 3 se inicia com a execução da tarefa **3.1 Configurar o ambiente de desenvolvimento,** a qual recebe o tutorial de instalação do JADE e JAMDER (disponível no Apêndice D) como entrada. O desenvolvedor deve configurar o ambiente de desenvolvimento de acordo com este tutorial antes de iniciar a codificação do SMA. Em seguida, a tarefa **3.2 Implementar o sistema multiagente com JAMDER** é executada, recebendo como artefatos de entrada o documento de requisitos final, o modelo de requisitos com iStar4RationalAgents e a modelagem do SMA em MAS-ML 2.0. A implementação deverá iniciar com a criação do código em JAMDER a partir do modelo de MAS-ML 2.0.

Em seguida na tarefa **3.3 Analisar rastreabilidade dos artefatos gerados** que recebe como artefatos de entrada o documento de requisitos final, a modelagem do SMA com MAS-ML 2.0, o modelo de requisitos com iStar4RationalAgents, o código da implementação feito em JAMDER e o checklist de rastreabilidade com o nível de projetos preenchido, os artefatos são submetidos a análise através do checklist com a finalidade de identificar possíveis inconsistências.

Em seguida, é tomada uma decisão com base na existência de problemas de rastreabilidade. Caso problemas tenham sido identificados, a execução do subprocesso continua com a tarefa **3.4 Corrigir problemas de rastreabilidade** que recebe o checklist com o nível de desenvolvimento preenchido. Nesta tarefa serão corrigidos os problemas identificados na tarefa 3.3 com base no checklist de rastreabilidade [Desenvolvimento] com o intuito manter a consistência e rastreabilidade entre os artefatos do projeto e a codificação

do SMA. Caso problemas de rastreabilidade não tenham sido identificados, o subprocesso é finalizado tendo gerado o código do SMA com JAMDER.

### 5.1.5 Testar Sistema Multiagente (Subprocesso 4)

Este subprocesso é iniciado após o fim da execução do subprocesso 3. Ele apresenta um conjunto de tarefas para auxiliar a realização de testes do SMA. O subprocesso pode ser visto na Figura 5.1.5 e é detalhado nos parágrafos a seguir.

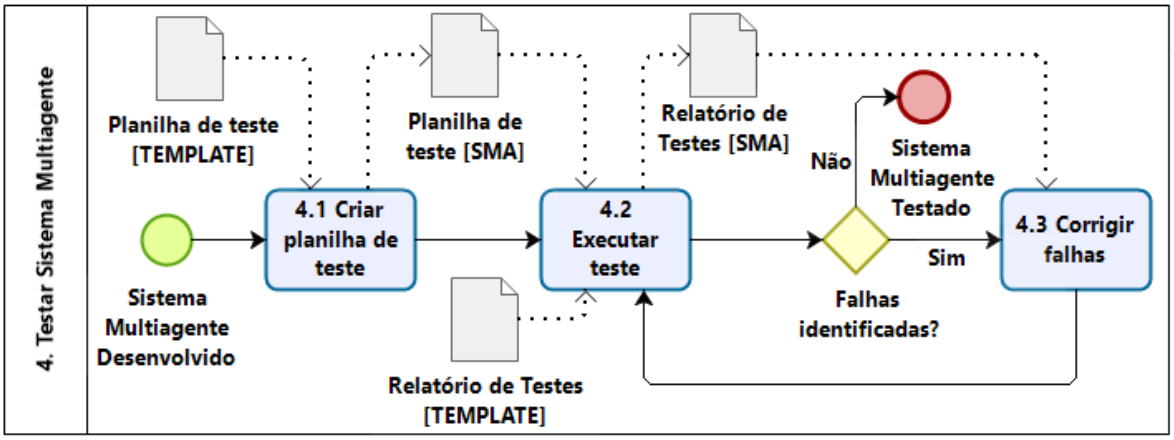



Figura 5.1.5 - Testar Sistema Multiagente.

O fluxo de testes se inicia com o SMA desenvolvido, este subprocesso possui três tarefas. A tarefa **4.1 Criar planilha de Teste** deste subprocesso consiste em preencher o template da planilha de testes (disponível no Apêndice E) para o SMA em desenvolvimento. Ao fim desta tarefa será gerado o documento da planilha de teste [SMA] que irá servir como artefato de entrada para a tarefa **4.2 Executar Teste**. Na tarefa 4.2 serão executados os testes de cada funcionalidade com base na planilha de testes. Após a execução dos testes, será gerado o Relatório de Testes com base no template de entrada (disponível no Apêndice F). O relatório de testes apresenta o resultado do teste de cada entidade do SMA.

Então é tomada uma decisão com base no relatório de testes. Caso sejam encontradas falhas, a tarefa **4.3 Corrigir Falhas** é iniciada, a qual recebe como artefato de entrada o relatório de testes. Nesta tarefa serão feitas as correções com base nas informações contidas no relatório de teste e então deverá retornar a tarefa 4.2. Quando não forem identificadas

falhas, este subprocesso é encerrado com o SMA testado. Consequentemente, o processo também é dado como finalizado após o encerramento do subprocesso de teste.

## 5.2 Ilustração do processo

O processo PSMAR foi definido por meio de engenharia reversa de um projeto de desenvolvimento de um SMA para o MOODLE (GONÇALVES, ARAUJO e CASTRO, 2019) e (GONÇALVES et al. 2014). Os resultados do desenvolvimento deste projeto foram apresentados por meio de artigos ao longo dos últimos anos. A modelagem de requisitos do SMA para o MOODLE foi apresentada em GONÇALVES, ARAUJO e CASTRO (2019). Já o projeto utilizando MAS-ML 2.0, a implementação com JAMDER e os testes foram executados no contexto de GONÇALVES et al. (2014). Assim, consideraremos este SMA para o MOODLE como a ilustração do PSMAR.

## REFERÊNCIAS

MORAIS II, M. **MAS-COMMONKADS+: Uma Extensão à Metodologia MAS-CommonKADS Para Suporte ao Projeto Detalhado De Sistemas Multiagentes Racionais**. 2010.

BITTENCOURT, G. **Inteligência artificial distribuída.** I workshop de computação do ITA, 1998.

CASTRO, J.; ALENCAR, F.; SILVA, C. T. L. L. **Engenharia de software orientada a agentes.** In: BREITMAN, K.; ANIDO, R. (Ed.). Atualizações em Informática, 2006.

CHELLA, A.; COSSENTINO, M.; SEIDITA, V. **Towards a methodology for designing artificial conscious robotic systems.** 2009. Disponível em: <http://www.aaai.org/ocs/index.php/FSS/FSS09/paper/view/882/1274>.

FREIRE, E. S.; GONÇALVES, E.; CORTÉS, M. I.; BRANDÃO, M. **TAO+: Extending the Conceptual Framework TAO to Support Internal Agent Architectures in Normative Multi-Agent Systems.** Electronic Notes in Theoretical Computer Science, v. 292, 2013.

DALPIAZ, F.; FRANCH, X; HORKOFF, J. **iStar 2.0 Language Guide**. arXiv:1605.07767, 2016 Disponível em: <https://arxiv.org/abs/1605.07767>

FRANKLIN, S.; GRAESSE, A. **Is it an agent, or just a program? a taxonomy for autonomous agents.** In: third international workshop on agents theories. [S.l.]: Springer-Verlag, 1996.

GONÇALVES, E.; OLIVEIRA, M. A.; FREIRES JUNIOR, J. H.; FEITOSA, G.; MENDES, D.; CORTÉS, M. I.; FEITOSA, R.; LOPES, Y. S. . **Uma Abordagem Baseada em Agentes de Apoio ao Ensino a Distância Utilizando Técnicas de Engenharia de Software**. X Simpósio Brasileiro de Sistemas de Informação, 2014.

GONÇALVES, E. J. T.; **MODELAGEM DE ARQUITETURAS INTERNAS DE AGENTES DE SOFTWARE UTILIZANDO A LINGUAGEM MAS-ML 2.0.** 2009

GONÇALVES, E.; ARAÚJO, J.; CASTRO, J. (2019). **iStar4RationalAgents: Modeling Requirements of Multi-agent Systems with Rational Agents**.

GONÇALVES, E. J. T., CORTES, M. I., CAMPOS, G. A. L., LOPES, Y. S., FREIRE, E. S., SILVA, V. T. D., ... & OLIVEIRA, M. A. D. (2015). **MAS-ML 2.0: supporting the modelling of multi-agent systems with different agent architectures.**

GONÇALVES, E. J. T.; CORTÉS, M. I.; CAMPOS G. A.; GOMES G. F.; DA SILVA V. T. **Towards the modeling reactive and proactive agents by using MAS-ML.** In: the 2010 ACM Symposium, 2010, Sierre. Proceedings of the 2010 ACM Symposium on Applied Computing - SAC ‘10. v. 2. p. 936-937

HENDERSON-SELLERS, B.; GIORGINI, P.; **Agent-Oriented Methodologies.** Idea Group Publishing. 2005.

IGLESIAS, C. A.; GARIJO, M. The agent-oriented methodology mas-commonkads. In: GIORGINI, B. H.-S. e P. (Ed.). Agented-Oriented Methodologies. [S.l.]: IDEA Group Publishing, p. 46–78, 2005.

JADE, Java Agent Development Framework, disponível em: http://jade.tilab.com/, acessado em: 7 de outubro de 2020.

JENNINGS, N. R. Coordination techniques for DAI. In: O'HARE, G.; JENNINGS, N. R. (Ed.). **Foundations of distributed artificial intelligence.** [S.l.]: John Wiley and Sons, 1996.

KITCHENHAM, B.; PFLEEGER, S. Principles of Survey Research, Software Engineering Notes, v. 27, n. 5, pp. 1- 20, 2002.

LOPES, Y.; CORTES, M.; GONÇALVES, E.; OLIVEIRA, R. (2018). **JAMDER: JADE to MULTI-Agent Systems Development Resource**. ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal.

Melo, J.; Sousa, A.; Agra, C.; Júnior, J.; Castro, J.;Alencar, F. **Formalization of mapping rules from istar to class diagram in UML**. 29th Brazilian Symposium on Software Engineering, 71-79, 2015.

MICHAELIS. **Agente**. Disponível em: <https://michaelis.uol.com.br/moderno-portugues/busca/portugues-brasileiro/agente/> Acesso em: 17 Abr. 2020.

PADGHAM, L.; WINIKOFF, M. **Prometheus: A practical agent-oriented methodology.** In: GIORGINI, B. H.-S. e P. (Ed.). Agented-Oriented Methodologies. [S.l.]: IDEA Group Publishing, 2005. p. 107–135.

RUSSELL, S.; NORVIG, P. **Artificial Intelligence A Modern Approach**. 3ª Edição. 1995. Prentice Hall, Singapore.

SOMMERVILLE. **Engenharia de Software**. 8ª Edição. 2007. Pearson Addison

WOOLDRIDGE, M. **An introduction to Multiagent Systems**. 2ª Edição. 2002. JOHN WILEY & SONS, LTD

SCHREIBER, G. et al. **Knowledge Engineering and Management: The CommonKADS Methodology.** [S.l.]: MIT Press, 2000.

Silva, V. T. **Uma linguagem de modelagem para sistemas multi-agentes baseada em um framework conceitual para agentes e objetos**. Tese de doutorado. Rio de Janeiro: PUC, Departamento de Informática, 2004.

# APÊNDICE A - Documento de Requisitos [TEMPLATE]

Este documento se destina a especificar os requisitos dos Sistema Multi-Agente (SMA) para o **<Domínio do Sistema>**. A descrição deste SMA é apresentada em três níveis. O nível 1 é uma descrição de alto nível apresentada de forma textual com intuito de servir de base para o detalhamento nos níveis posteriores. O nível 2 é especificado tomando como base a descrição apresentada no nível 1, neste nível são detalhadas as entidades que compõem o SMA com seus tipos e descrições. Finalmente, o nível 3 é especificado tomando como base o que foi definido nos níveis 1 e 2, detalhando assim, os elementos contidos internamente em cada uma das entidades e seus relacionamentos. O nível 1 é especificado por meio de descrição textual livre, enquanto os níveis 2 e 3 são especificados por meio de tabelas.

**NÍVEL 1 -** No nível 1 o sistema deve ser descrito textualmente por meio do preenchimento dos campos que em uma abordagem inicial, deve ser coletada uma **especificação geral** do SMA. Descreva em alto nível os pontos a seguir para que facilite a compreensão do sistema de modo geral:

1. Qual o domínio do sistema multiagente?
2. O SMA será de simulação ou acoplado a um sistema real?
3. Quais os principais objetivos do sistema e como alcançá-los?
4. No caso de um multiagente que executará acoplado a um sistema real, como será a interação entre os agentes e o sistema?
5. Quais as entidades envolvidas e a relação entre elas?

**NÍVEL 2 -** No nível 2 as entidades do sistema devem ser identificadas após o preenchimento dos campos do nível 1 na abordagem inicial, deve ser coletada uma descrição específica do SMA.

| **Ambiente** | | | |
|---|---|---|---|
| **ID** | **Nome** | **Descrição** | **Tipo** |
| AMB-001 | <Nome do ambiente> | <Descrição do ambiente> | <Tipo do ambiente> |

| **Organização** | | |
|---|---|---|
| **ID** | **Nome** | **Descrição** |
| ORG-001 | <Nome da Organização> | <Descrição da Organização> |

| Agente | | | |
|---|---|---|---|
| **ID** | **Nome** | **Descrição** | **Tipo** |
| Ag-001 | <Nome do Agente> | <Descrição do Agente> | <Tipo do Agente> |

| Papéis de Agentes | | | |
|---|---|---|---|
| **ID** | **Nome** | **Descrição** | **Tipo** |
| PA-001 | <Nome do papel> | <Descrição do papel> | <Tipo do papel> |

Relacionamentos (Como conectar as entidades do SMA entre si)

| Entidade fonte | Relacionamento | Entidade alvo |
|---|---|---|
| <Nome da entidade fonte> | <Tipo de relacionamento entre as entidades> | <Nome da entidade alvo> |

**Nível 3 -** No nível 3, as entidades do sistema identificadas após o preenchimento dos campos do nível 2, devem ser coletadas **especificações detalhadas** das entidades do SMA.

| **<Nome da Entidade>** | |
|---|---|
| **<Tipo do Elemento interno>** | |
| **Nome** | **Descrição** |
| <Nome do elemento interno> | <Descrição do elemento interno> |

| **<Nome da Entidade>** | | | |
|---|---|---|---|
| **<Tipo do Elemento interno>** | | **<Tipo do Elemento interno>** | |
| **Nome** | **Descrição** | **Nome** | **Descrição** |
| <Nome do elemento interno> | <Descrição do elemento interno> | <Nome do elemento interno> | <Descrição do elemento interno> |

Relacionamentos (Como conectar as entidades do sma entre si)

| **<Nome da Entidade>** | | |
|---|---|---|
| **Entidade fonte** | **Relacionamento** | **Entidade alvo** |
| <Nome da Entidade fonte> | <Tipo de Relacionamento entre as entidades> | <Nome da Entidade alvo> |

| **<Nome da Entidade>** | | |
|---|---|---|
| **Entidade fonte** | **Relacionamento** | **Entidade alvo** |
| <Nome da Entidade fonte> | <Tipo de Relacionamento entre as entidades> | <Nome da Entidade alvo> |
| <Nome da Entidade fonte> | <Tipo de Relacionamento entre as entidades> | <Nome da Entidade alvo> |

## Modelo de requisitos com iStar4RationalAgents

*Aqui deve conter as imagens dos modelos SD e SR de iStar.*

## APÊNDICE B - Planilha de Mapeamento de Construtores iStar4RationalAgents e MAS-ML 2.0

| Elementos iStar4RationalAgents | Elemento MAS-ML 2.0 |
|---|---|
| Objetivo | Objetivo |
| Crença | Crença |
| Agente reativo simples | Agente reativo simples |
| Agente reativo baseado em modelo | Agente reativo baseado em modelo |
| Agente baseado em objetivos | Agente baseado em objetivos |
| Agente baseado em utilidade | Agente baseado em utilidade |
| Papel do agente reativo simples | Papel do agente reativo simples |
| Papel do agente reativo baseado em modelo | Papel do agente reativo baseado em modelo |
| Papel do agente baseado em objetivos | Papel do agente baseado em objetivos |
| Papel do agente baseado em utilidade | Papel do agente baseado em utilidade |
| Função próximo | Função próximo |
| Função formulação de objetivo | Função formulação de objetivo |
| Função formulação de problema | Função formulação de problema |
| Função utilidade | Função utilidade |
| Ação | Ação |
| Dever | Dever |
| Direito | Direito |
| Causa-efeito | Representado junto das ações dos agentes reativos |
| Percepção | Percepção |
| Planejamento | Planejamento |
| Organização | Organização |
| Ambiente | Ambiente |
| Plano | Plano |
| Agentes, Papéis | Classe |
| O relacionamento FAZ PARTE DE entre posições, agentes ou funções. | Agregação de classes. |
| Relacionamento É UM entre posições, agentes ou funções. | Generalização / especialização de classe. |

| | |
|---|---|
| O relacionamento *OCCUPIES* entre um agente e uma posição. | Associação de classe chamada *OCCUPIES*. |
| O relacionamento *COVERS* entre uma posição e uma função. | Associação de classe chamada *COVERS*. |
| Relacionamento *PLAYS* entre um agente e uma função. | Associação de classe chamada *PLAY*S. |
| Tarefas definidas no modelo SD. | Métodos com visibilidade pública. |
| Tarefas definidas no modelo SR. | Métodos com visibilidade privada. |
| Recursos definidos no modelo SD. | Classe *IF THIS* dependência tem características de um objeto. |
| Recursos definidos no modelo SD. | Atributo com visibilidade privada em classe que representa o ator dependente se esta dependência não puder ser caracterizada como um objeto |
| Recursos (sub-recursos) definidos no modelo SR. | Atributo com visibilidade privada na classe que representa o ator ao qual pertence o sub-recurso (se este sub-recurso não puder ser entendido como um objeto). |
| Recursos (sub-recursos) definidos no modelo SR. | Uma classe independente, caso contrário. |
| Metas (leves) no modelo SD. | Atributo com visibilidade pública na classe que representa o dependente. |
| Metas (leves) no modelo SD. | Atributo com visibilidade pública na classe que representa o ator ao qual pertence o sub objetivo. |
| Decomposição de tarefas. | Representado por pré e pós-condições (expressas em OCL) da operação pUML correspondente. |
| *Goal/Quality* | A disjunção dos valores médios implica o valor final. |
| Objetivo (*Quality*) - Tarefa, Recurso-Tarefa. | A pós-condição da tarefa dos meios implica o valor do fim. |
| Tarefa-Tarefa | A disjunção da pós-condição dos meios implica nas pós-condições do fim. |
| *Needed-by* | Composição |
| *Qualification* | Associação |
| *Dependency* | Dependency |

## APÊNDICE C - Checklist de Rastreabilidade [TEMPLATE]

Este documento se destina a fazer uma análise de rastreabilidade entre as fases do processo PSMAR para checar a uniformidade do sistema multiagente em questão. Na primeira tabela deverão ser assinaladas as fases do processo que já foram concluídas e que os artefatos estão em conformidade. Na segunda tabela deverão ser assinalados as entidades em conformidade em cada fase do processo.

| **Quais subprocessos já foram concluídos?** | | | |
|---|---|---|---|
| [ ] Levantamento e especificação de Requisitos do SMA | [ ] Projeto do SMA com MAS-ML 2.0 | [ ] Desenvolvimento do SMA com JAMDER | [ ] Teste do SMA |

| **Entidades** | **Levantamento e especificação de Requisitos do SMA** | | **Projeto do SMA com MAS-ML 2.0** | **Desenvolvimento do SMA com JAMDER** | **Teste do SMA** |
|---|---|---|---|---|---|
| | **Documento de Requisitos** | **Modelo de requisitos com iStar4RationalAgents** | | | |
| <Entidade-01> | [ ] | [ ] | [ ] | [ ] | [ ] |
| <Entidade-02> | [ ] | [ ] | [ ] | [ ] | [ ] |

# APÊNDICE D - Tutorial de instalação do JADE e JAMDER

**Requisitos:**

1. Windows 10
2. Eclipse IDE for Java EE Developers

**Nota:** Ao executar o arquivo de instalação do Eclipse poderão ser apresentadas outras versões do programa, esteja certificado de que irá instalar a versão **Eclipse IDE for Java Developers**.

A ferramenta será utilizada para CODIFICAR e não para MODELAR, caso seja instalado outra versão do programa os passos a seguir não irão funcionar.

3. Jade - JAVA Agent DEvelopment Framework
   3.1. Acessar o site oficial do JADE onde o arquivio estará disponível para download: https://jade.tilab.com/
   3.2. Ao clicar no link acima você será redirecionado para a página inicial, passe o mouse sobre “Download” e clique em “Jade”.


   3.3. Após aceitar os termos, clique em:

| JADE | ~ File size | Description of the content |
|---|---|---|
| jadeAll | 17.7 MB | This file contains all JADE, i.e. it is just composed of the 4 files below. If it is too large for downloading, the 4 files below might be downloaded instead. |
| jadeBin | 2.5 MB | This file contains JADE already compiled and ready to be used, i.e. a set of JAVA archive JAR files. |
| jadeDoc | 12.8 MB | This file contains all the JADE documentation included the Administrator's Guide and and the Programmer's Guide. NOTICE THAT all the documentation is also available on-line. |
| jadeSrc | 2.3 MB | This file contains all the JADE source code. |
| jadeExamples | 490 KB | This file contains the source code of the examples and a simple demo. All the examples and demo must be compiled. |

   3.4. Após baixar o JADE, você precisará baixar o JAMDER. O arquivo está disponível no link abaixo:
      3.4.1. https://drive.google.com/file/d/1_gR8Y1WHMhgdWQn7hG6DjYHPOb1NIYTj/view?usp=sharing

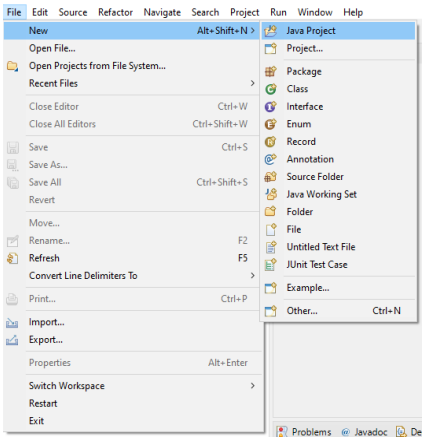
3.5. Salve na pasta de Downloads ou em uma pasta de sua preferência.

**Saída:**


**Instalação**

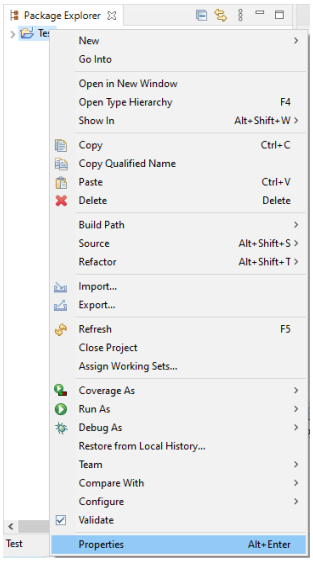
1. Instalar o Eclipse;
2. Após finalizar o processo de instalação do Eclipse, com o programa em execução crie um novo projeto Java:

   2.1. File -> New -> Java Project

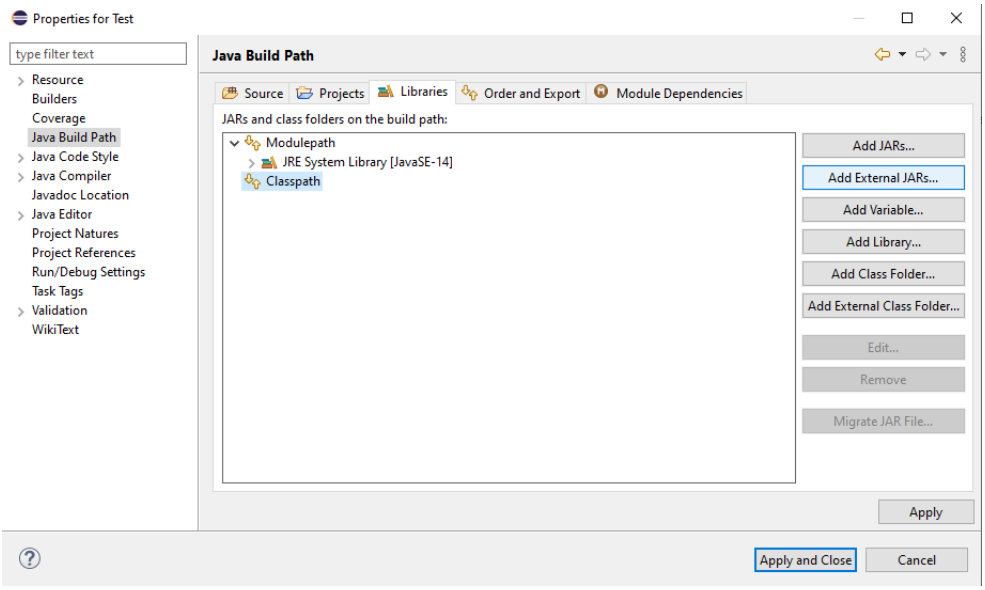



3. Após criar o projeto é necessário importar JADE-all-*.zip da seguinte forma:

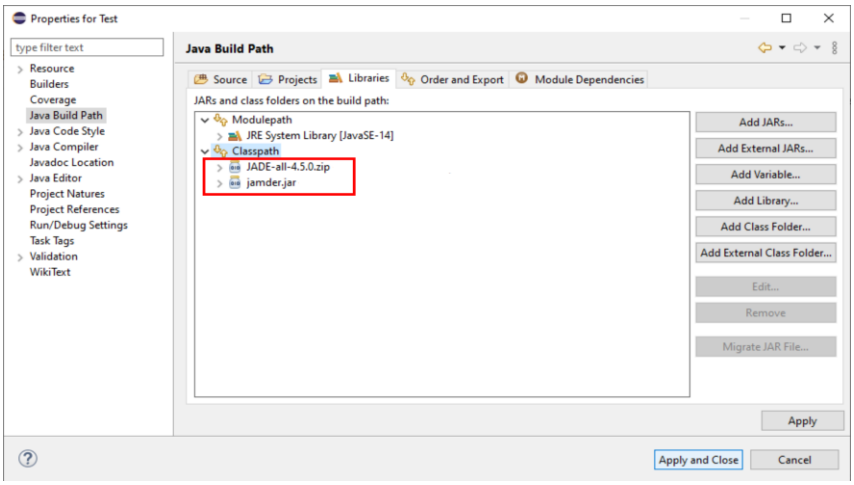
   3.1. Clicar com o botão direito do mouse sobre o nome do projeto e em seguida clicar em “Propriedades” (ou *properties*);



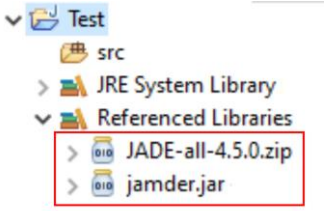

4. Após o passo anterior, você irá clicar em **Java Build Path** -> **Libraries** -> **Classpath** -> **Add External JARs**




5. Então você deverá selecionar o arquivo JADE-all-*.zip, o mesmo procedimento deverá ser feito para adicionar o arquivo jamder.jar.




6. Após selecionar os dois arquivos você irá clicar em **Apply and Close**.
7. Ao final da execução deste tutorial a saída deverá ser a seguinte:
   7.1. JADE e jamder adicionados como “Bibliotecas Referenciadas” (*Referenced Libraries*)



# APÊNDICE E - Planilha de Teste [TEMPLATE]

| **Planilha de Teste** | |
|---|---|
| Caso de Uso: | <Nome do Agente/Entidade> |
| Complexidade: | <Nível da complexidade> |
| Funcionalidades: | <Funcionalidade do Agente/Entidade> |
| Pré-Condições: | <Condições para que as funcionalidades sejam satisfeitas> |
| Observações: | <Observações a serem feitas sobre algo do Agente/Entidade> |
| **<Funcionalidade 01>** | |
| **Objetivo da <Funcionalidade>: <Descrição do objetivo da funcionalidade>** | |
| **Funcionamento da <Funcionalidade>: <Descrição do processo de execução da funcionalidade>** | |
| **Pré-requisito para que o objetivo seja alcançado: <Listagem de pré-requisitos que são necessário para que o objetivo seja alcançado>** | |
| **1) Possíveis resultados do Teste:** | |
| **<Cenário da funcionalidade>** | <Descrição do cenário> |
| **<1º Caso>** | <Descrição do 1º caso> |
| **<2º Caso>** | <Descrição do 2º caso> |

| <Funcionalidade 01> | |
|---|---|
| **Dados Válidos** | |
| **1) Casos de Teste:** | **2) Ideias de Teste:** |
| 1) Happy Day - <Nome do caso de teste> | <Ideia de teste 1> |
| | <Ideia de teste 2> |
| **Dados Inválidos** | |
| **1) Casos de Teste:** | **2) Ideias de Teste:** |
| 1) Erro - <Nome do caso de teste> | <Ideia de teste 1> |
| | <Ideia de teste 2> |

## APÊNDICE F - Relatório de Teste [TEMPLATE]

Este documento apresenta o relatório de teste das entidades do sistema multiagente desenvolvido para *<finalidade do sma>*. Na tabela estão listadas as entidades do sistema multiagente, suas funcionalidades (comportamentos) testados e o resultado de cada um deles (sucesso ou falha).

| **<Entidade 01>** | | | |
|---|---|---|---|
| **Funcionalidade** | **Resultado** | | **Descrição do erro** |
| <Fun 01> | Falha [ ] | Sucesso [ ] | |
| <Fun 02> | Falha [ ] | Sucesso [ ] | |
| <Fun 03> | Falha [ ] | Sucesso [ ] | |
| <Fun 04> | Falha [ ] | Sucesso [ ] | |
| <Fun 05> | Falha [ ] | Sucesso [ ] | |

| **<Entidade 02>** | | | |
|---|---|---|---|
| **Funcionalidade** | **Resultado** | | **Descrição do erro** |
| <Fun 01> | Falha [ ] | Sucesso [ ] | |
| <Fun 02> | Falha [ ] | Sucesso [ ] | |
| <Fun 03> | Falha [ ] | Sucesso [ ] | |
| <Fun 04> | Falha [ ] | Sucesso [ ] | |
| <Fun 05> | Falha [ ] | Sucesso [ ] | |

# APÊNDICE G - Tutorial de instalação do Plugin MAS-ML Tool

**Requisitos:**

1. Windows 10
2. Eclipse Modeling Tools

(http://www.eclipse.org/downloads/packages/eclipse-modeling-tools/oxygen1)

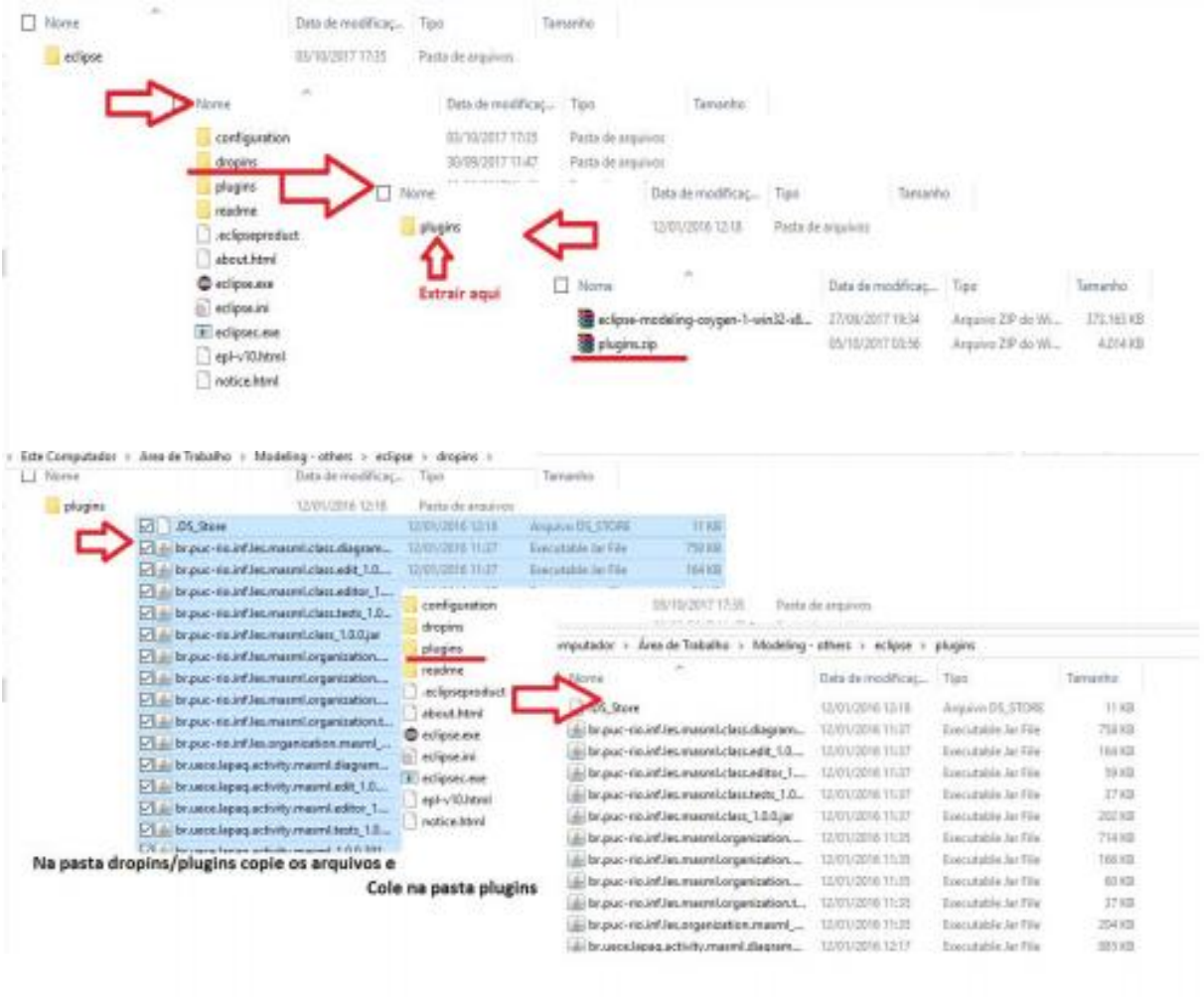

Nota: Deve ser baixado o Eclipse Modeling Tools, NÃO o Eclipse IDE ou outras versões. Vamos MODELAR e não CODIFICAR, então deve ser usado/baixado o Eclipse Modeling Tools , sendo assim se for utilizado o Eclipse IDE for Java EE Developers, ou qualquer outra que não seja a primeira (Eclipse Modeling Tools), os plugins da MAS-ML NÃO irão funcionar.

3. Plugins de MAS-ML tool

(https://drive.google.com/file/d/0B3o24ViY1B22Q2hsajdnNTlMTmM/view?usp=drives dk)

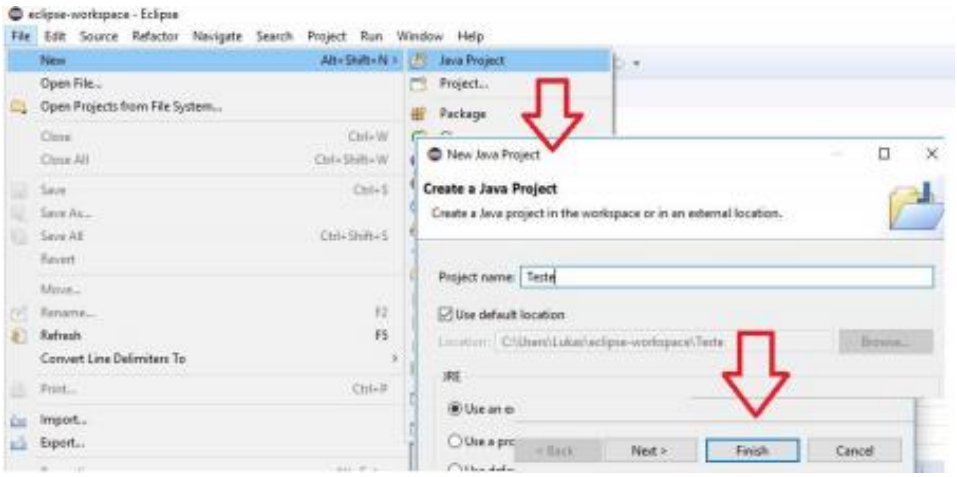
**Instalação:**

1. Extrair o eclipse;
2. Extrair o .zip do Plugins MAS-ML tool baixado;;
2. Copiar o conteúdo de MAS-ML tool extraído para a pasta **dropins/plugins** do eclipse;

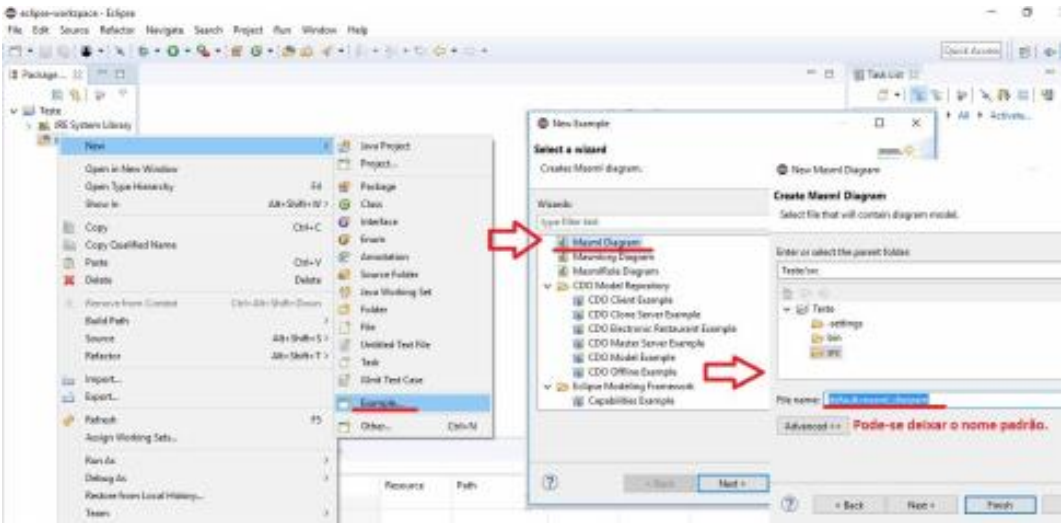



**Testar funcionamento do Plugin:**

1. Inicie o eclipse
2. New>Java Project -- Nome do projeto(*Neste exemplo é Teste*) e Finish




3. Clique direito na pasta src>New>Example>Masml Diagram>Finish




4. Agora vamos inserir uma classe