



UNIVERSIDADE FEDERAL DE MATO GROSSO
INSTITUTO DE COMPUTAÇÃO
COORDENAÇÃO DE ENSINO DE GRADUAÇÃO EM
CIÊNCIA DA COMPUTAÇÃO

**RELATÓRIO DE ESTÁGIO SUPERVISIONADO
DESENVOLVIMENTO DE UMA FERRAMENTA DE
MODELAGEM PARA SISTEMAS MULTIAGENTES
CULTURAIS**

IGOR HIDEKI TRINDADE

CUIABÁ – MT

2014

UNIVERSIDADE FEDERAL DE MATO GROSSO
INSTITUTO DE COMPUTAÇÃO
COORDENAÇÃO DE ENSINO DE GRADUAÇÃO EM
CIÊNCIA DA COMPUTAÇÃO

**RELÁTORIO DE ESTÁGIO SUPERVISIONADO
DESENVOLVIMENTO DE UMA FERRAMENTA DE
MODELAGEM PARA SISTEMAS MULTIAGENTES
CULTURAIS**

IGOR HIDEKI TRINDADE

Relatório apresentado ao Instituto de
Computação da Universidade Federal de
Mato Grosso, para obtenção do título de
Bacharel em Ciência da Computação.

CUIABÁ – MT

2014

UNIVERSIDADE FEDERAL DE MATO GROSSO
INSTITUTO DE COMPUTAÇÃO
COORDENAÇÃO DE ENSINO DE GRADUAÇÃO EM
CIÊNCIA DA COMPUTAÇÃO

IGOR HIDEKI TRINDADE

Relatório de Estágio Supervisionado apresentado à Coordenação do Curso de Ciência da Computação como uma das exigências para obtenção do título de Bacharel em Ciência da Computação da Universidade Federal de Mato Grosso

Aprovado por:

Prof. MSc. DANIEL AVILA VECCHIATO
Instituto de Computação

Prof. MSc. FERNANDO CASTILHO
Instituto de Computação

Prof. MSc. KAREN DA SILVA FIGUEIREDO
Instituto de Computação

DEDICATÓRIA

Aos meus pais pelo apoio e confiança

À minha irmã pelo desvelo

AGRADECIMENTOS

Agradeço à Deus por tornar possível a realização deste trabalho, por me fortalecer perante as dificuldades encontradas em meu caminho.

Agradeço à minha família pela minha formação de caráter e pelo seu grande esforço a fim de proporcionar-me esta oportunidade.

Agradeço a todos meus professores do Instituto de Computação que transmitiram seu conhecimento e se esforçaram para enriquecer minha formação acadêmica e profissional.

Agradeço em especial a professora Karen da Silva Figueiredo, por me auxiliar e instruir, sanando sempre minhas dúvidas com paciência na realização deste trabalho.

Agradeço a todos meus amigos que fiz no Instituto de Computação por me proporcionarem esta experiência única, por tornarem especial esta etapa que passou e que vão continuar presentes em minha vida.

Agradeço em especial aos meus amigos que conheci no curso de Ciência da Computação, Adrian Procopiou e Sidnei Polo pelo apoio, incentivo e divertimento, por fazerem parte da minha formação e ajudarem sempre que precisei.

Finalmente, agradeço a todos os meus amigos que já me acompanhavam antes desta etapa e também aqueles que conheci fora da faculdade ao decorrer destes anos. Sou grato pelas risadas, conselhos e diversão, por confiarem em mim, escutarem todas minhas reclamações e mesmo assim estarem ao meu lado dispostos a me amparar.

SUMÁRIO

LISTA DE FIGURAS	7
LISTA DE TABELAS.....	8
LISTA DE SIGLAS E ABREVIATURAS	9
RESUMO	10
INTRODUÇÃO	11
1. REVISÃO DE LITERATURA	13
1.1. AGENTES E SISTEMAS MULTIAGENTES.....	13
1.2. ONTOLOGIAS	14
1.2.1. <i>Ontologia</i>	14
1.2.2. <i>Ontologia para SMA Culturais</i>	14
1.3. MODELAGEM, MODELO E METAMODELO.....	18
2. MATERIAIS, TÉCNICAS E MÉTODOS.....	19
2.1. A FERRAMENTA E AMBIENTE ECLIPSE	19
2.2. LINGUAGEM JAVA	20
2.3. JAVA DEVELOPMENT KIT (JDK)	21
2.4. GRAPHICAL MODELING FRAMEWORK (GMF)	21
2.5. DESENVOLVIMENTO	22
2.5.1. <i>Criando o projeto GMF</i>	22
2.5.2. <i>Criando o metamodelo</i>	24
2.5.3. <i>Criando modelo gráfico</i>	28
2.5.4. <i>Criando a paleta gráfica</i>	30
2.5.5. <i>Criando modelo de mapeamento</i>	32
2.5.6. <i>Criando modelo editor de diagramas e gerando o projeto executável</i>	33
3. RESULTADOS	35
4. DIFICULDADES ENCONTRADAS	41
5. CONCLUSÕES.....	42
6. REFERÊNCIAS BIBLIOGRÁFICAS.....	43

LISTA DE FIGURAS

FIGURA 1 - AGENTE E AMBIENTE	13
FIGURA 2 - ONTOLOGIA DE SMA CULTURAIS	17
FIGURA 3 - EXEMPLO DE METAMODELO	18
FIGURA 4 - TELA PRINCIPAL DA IDE ECLIPSE	20
FIGURA 5 - TELA GMF DASHBOARD NO ECLIPSE	21
FIGURA 6 - TELA DE COMO CRIAR UM PROJETO GMF NO ECLIPSE	22
FIGURA 7 - TELA DE SELEÇÃO DO PROJETO GMF	23
FIGURA 8 - TELAS DE DEFINIÇÃO DE NOME DO PROJETO GMF E EXIBIÇÃO DA GMF DASHBOARD	24
FIGURA 9 - TELA DE CRIAÇÃO DO MODELO DE DOMÍNIO	24
FIGURA 10 - TELA DE DEFINIÇÃO DAS PROPRIEDADES DO MODELO DE DOMÍNIO	25
FIGURA 11 - TELA DE COMO CRIAÇÃO DE CLASSES NO MODELO DE DOMÍNIO	25
FIGURA 12 - TELA DE DEFINIÇÃO DAS PROPRIEDADES DAS CLASSES NO MODELO DE DOMÍNIO	26
FIGURA 13 - TELA DA ABA DE PROPRIEDADES DAS CLASSES DE UM MODELO DE DOMÍNIO	26
FIGURA 14 - TELA DA ABA DE PROPRIEDADES DOS ATRIBUTOS DE UM MODELO DE DOMÍNIO	26
FIGURA 15 - TELA DA ABA DE PROPRIEDADES DOS RELACIONAMENTOS DE UM MODELO DE DOMÍNIO	27
FIGURA 16 - TELA DE COMO GERAR O DIAGRAMA REFERENTE AO MODELO DE DOMÍNIO	27
FIGURA 17 - TELA DE COMO CRIAR UM MODELO DE GERAÇÃO DE CÓDIGO NO DASHBOARD DO GMF	28
FIGURA 18 - TELA DE COMO GERAR OS CÓDIGOS DAS CLASSES NO MODELO DE GERAÇÃO DE CÓDIGO	28
FIGURA 19 - TELA DE DEFINIÇÃO GRÁFICA DAS CLASSES	29
FIGURA 20 - TELA DE CONFIGURAÇÃO DO MODELO GRÁFICO	30
FIGURA 21 - TELA DE SELEÇÃO DE QUAIS ELEMENTOS SERÃO EXIBIDOS NA PALETA	31
FIGURA 22 - TELA DE CONFIGURAÇÃO DO MODELO DA PALETA	31
FIGURA 23 - TELA DE CONFIGURAÇÃO E ASSOCIAÇÃO DOS ELEMENTOS AS CLASSES	32
FIGURA 24 - TELA DE CONFIGURAÇÃO DO MODELO DE MAPEAMENTO	33
FIGURA 25 - TELA DE CRIAÇÃO DO MODELO EDITOR DE DIAGRAMAS NO GMF <i>DASHBOARD</i>	33
FIGURA 26 - TELA DA GERAÇÃO DO PROJETO EXECUTÁVEL DA FERRAMENTA	34
FIGURA 27 - EXEMPLO DE MODELAGEM NA FERRAMENTA OBTIDA NESTE TRABALHO	35
FIGURA 28 - EXEMPLO DOS ELEMENTOS, ARESTAS E RÓTULOS	38
FIGURA 29 - PALETA DA FERRAMENTA DE MODELAGEM DE SMA CULTURAIS	39
FIGURA 30 - EXEMPLO DE UMA MODELAGEM DE UM SMA CULTURAL	40

LISTA DE TABELAS

TABELA 1 - ENTIDADES DOS MODELOS DE SMA CULTURAIS.....	36
TABELA 2 - RELACIONAMENTOS DOS MODELOS DE SMA CULTURIAS	37

LISTA DE SIGLAS E ABREVIATURAS

API	<i>Application Programming Interface</i> – Interface de Programação de Aplicações
EMF	<i>Eclipse Modeling Framework</i> – Framework de Modelagem do Eclipse
GEF	<i>Graphical Editing Framework</i> – Framework de Edição Gráfica
GMF	<i>Graphical Modeling Framework</i> – Framework de Modelagem Gráfica
IDE	<i>Integrated Development Environment</i> – Ambiente de Desenvolvimento Integrado
JDK	<i>Java Development Kit</i> – Kit de Desenvolvimento Java
JRE	<i>Java Runtime Environment</i> – Ambiente de Tempo de Execução Java
SMA	Sistemas Multiagentes
SO	Sistema Operacional

RESUMO

Na área de agentes em computação há um novo tema que vem sendo trabalhado denominado SMA Cultural, ele permite que os agentes em seu sistema sejam mais autônomos, i. e. as ações dos agentes são mais amplas para estabelecer uma ordem social entre os mesmos, a fim de fazer o sistema dirigir-se a uma solução desejada, mas sem predefinir as especificações das ações de seus agentes. Este trabalho propõe o desenvolvimento de uma ferramenta de modelagem que permita a criação de modelos de SMA Culturais por analistas de sistemas, pois até o presente momento não existem ferramentas com esta finalidade. Ferramentas de modelagem tem como finalidade manter a organização e o cumprimento dos requisitos na etapa de desenvolvimento do software. Os modelos auxiliam a ter uma visão mais abrangente do funcionamento do sistema. Pretende-se também que através da elaboração deste trabalho o aluno em estágio aprenda os conceitos relacionados a agentes, SMA, ontologias e modelagem, bem como a trabalhar com as ferramentas: IDE Eclipse, Linguagem de programação Java, *Java Development Kit* (JDK) e *Graphical Modeling Framework* (GMF).

INTRODUÇÃO

Em computação, agente é um software ou parte de um software capaz de perceber o que ocorre no ambiente em que está situado, tendo autonomia para realizar um repertório de possíveis ações, podendo assim alterar o ambiente em que habita. O sistema que possui um ou mais agentes é chamado de Sistema Multiagente (SMA). Em um SMA, os agentes nele contidos interagem entre si podendo colaborar, cooperar ou competir para alcançar um objetivo em comum (BORDINI et al., 2007).

Assim como o Paradigma Orientado a Objetos, com os SMA surge o Paradigma Orientado a Agentes e junto com ele a necessidade de desenvolvimento de ferramentas para apoio às atividades de análise, projeto, desenvolvimento e teste desse tipo de sistema (JENNINGS, 1999).

O Paradigma de Programação Orientado a Agentes está passando por vários desafios e tem ganhado cada vez mais espaço, tanto no mercado de trabalho como no meio acadêmico. Este paradigma tem como foco o desenvolvimento de software segundo o estado dos agentes e vem ganhando espaço em áreas como: controle e automação, telecomunicações, sistemas de transporte, gerenciamento de informações, e-commerce e jogos interativos.

Com o crescimento deste paradigma temos como consequência o aumento na quantidade de ferramentas voltadas para engenharia de software orientadas a agentes. Ferramentas de modelagem tem como finalidade manter a organização e o cumprimento dos requisitos na etapa de desenvolvimento do software. Os modelos auxiliam a ter uma visão mais abrangente do funcionamento do sistema, convertendo uma visão da realidade em uma representação dela (VAN GIGCH, 1991).

Na área de agentes há um novo tema que vem sendo trabalhado denominado SMA Cultural, ele permite que os agentes em seu sistema sejam mais autônomos, i. e. as ações dos agentes são mais amplas para estabelecer uma ordem social entre os mesmos, a fim de fazer o sistema dirigir-se a uma solução desejada, mas sem predefinir as especificações das ações de seus agentes (MARQUES e FIGUEIREDO, 2014).

Desta forma, o objetivo deste trabalho é desenvolver uma ferramenta de modelagem que permita a criação de modelos de SMA Culturais por analistas de sistemas, pois até o presente momento não existem ferramentas com esta finalidade.

Pretende-se ainda que, através da elaboração deste trabalho o aluno em estágio aprenda os conceitos relacionados a agentes, SMA, ontologias e modelagem, bem como a trabalhar com as ferramentas: IDE Eclipse, Linguagem de programação Java, *Java Development Kit* (JDK) e *Graphical Modeling Framework* (GMF).

A partir desta introdução, este trabalho está organizado da seguinte forma: no Capítulo 1 é apresentada a revisão de literatura com as definições de agentes e sistemas multiagentes, a definição de ontologia e a ontologia de SMA Culturais utilizada para a criação da ferramenta e também as definições de modelagem, modelo e metamodelo; no Capítulo 2 são apresentados os materiais, técnicas e métodos utilizados na realização deste trabalho, tais como: IDE Eclipse, Linguagem de programação Java, *Java Development Kit* (JDK) e *Graphical Modeling Framework* (GMF), no Capítulo 3 são apresentados os resultados atingidos e no Capítulo 4 são apresentadas as dificuldades encontradas no decorrer deste trabalho. Por fim, no Capítulo 5 é apresentada a conclusão obtida e no Capítulo 6 as referências bibliográficas utilizadas como apoio a este trabalho.

1. REVISÃO DE LITERATURA

1.1. Agentes e Sistemas Multiagentes

Em computação, um agente é um software ou parte de um software capaz de perceber o que ocorre no ambiente em que está situado, tendo autonomia em realizar um conjunto de possíveis ações, podendo assim alterar o ambiente em que habita (BORDINI et al., 2007) (Figura 1). As ações realizadas pelos agentes podem ser muitas vezes feitas a favor de uma pessoa ou de outro agente, para isso os agentes podem: envolver tarefas automatizadas, usar alguma inteligência, comunicar com o usuário ou com outros agentes de forma cooperativa e coordenada, aprender e mudar o seu comportamento ao longo do tempo, e operar por sua própria iniciativa (OMG AGENT PLATFORM SPECIAL INTEREST GROUP, 2011).

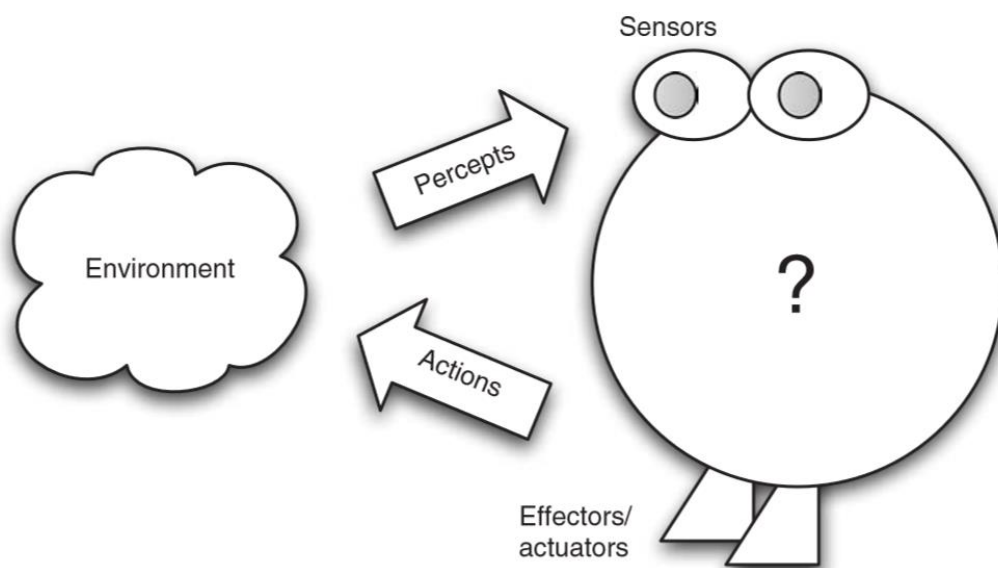


Figura 1 - Agente e ambiente
Fonte: BORDINI et al., 2007

A Figura 1 ilustrada por BORDINI et al. (2007) demonstra uma arquitetura genérica de Agentes em que as ações podem ser realizadas dentro do ambiente pelos atuadores do agente de acordo com a percepção realizada através de sensores do mesmo.

O sistema que possui um ou mais agentes é chamado de Sistema Multiagente (SMA). Em um SMA, os agentes nele contidos interagem entre si podendo colaborar, cooperar ou competir para alcançar um objetivo em comum (BORDINI et al., 2007).

Os agentes não são as únicas entidades em um SMA, dentro desse sistema é possível a existência de entidades como: organizações, ambientes, papéis, crenças e várias outras que serão vistas nas próximas seções.

Assim como o Paradigma Orientado a Objetos, com os SMA surge o Paradigma Orientado a Agentes e junto com ele a necessidade de desenvolvimento de ferramentas para apoio às atividades de análise, projeto, desenvolvimento e teste desse tipo de sistema (JENNINGS, 1999). O Paradigma de Programação Orientado a Agentes está passando por vários desafios e tem ganhado cada vez mais espaço, tanto no mercado de trabalho como no meio acadêmico. Este paradigma tem como foco o desenvolvimento de software segundo o estado dos agentes e vem ganhando espaço em áreas como: controle e automação, telecomunicações, sistemas de transporte, gerenciamento de informações, e-commerce e jogos interativos.

1.2. Ontologias

1.2.1. Ontologia

Uma ontologia é um artefato projetado para fornecer potenciais termos para descrever conceitos e relações que existem em um domínio específico do conhecimento (real ou imaginado) de interesse (GRUBER, 1993).

A ontologia é o núcleo de qualquer sistema de representação do conhecimento em um determinado domínio, sua análise tem como objetivo esclarecer a estrutura de conhecimento.

Análises ontológicas fracas podem levar a bases de conhecimentos incoerentes, a forma fundamental para evitar essas incoerências na concepção de um sistema de conhecimento é fazer uma análise eficaz (CHANDRASEKARAN, 1999).

1.2.2. Ontologia para SMA Culturais

Um SMA Cultural permite que os agentes em seu sistema sejam mais autônomos, i. e. as ações dos agentes são mais amplas para estabelecer uma ordem social entre os mesmos, a fim de fazer o sistema dirigir-se a uma solução desejada, mas sem predefinir as especificações das ações de seus agentes (MARQUES e FIGUEIREDO, 2014).

“Um sistema normativo, ou seja, cultural, se esforça em estabelecer convenções para seus membros. Isto é vantajoso, pois aprendizagem social permite os

indivíduos a imitar e aprender com outros, poupando-os do processo de teste e erro” (BOYAD, 1985, apud MARQUES e FIGUEIREDO, 2014).

Desta forma Marques e Figueiredo (2014) elaboraram uma ontologia de domínio para SMA Culturais que é utilizada como base para a elaboração da ferramenta proposta nesse trabalho. A ontologia de Marques e Figueiredo (2014) está ilustrada na Figura 2, e seus conceitos e relacionamentos são descritos pelos autores conforme os itens abaixo:

Environment (Ambiente): ambiente em que as organizações e agentes do sistema habitam. Os ambientes podem possuir normas para restringir o comportamento dos agentes do ambiente.

Organization (Organização): organização ou grupo de agentes. Uma organização pode ser composta de suborganizações, neste caso as suborganizações desempenham um papel na superorganização. As organizações possuem uma cultura e podem descrever normas para serem seguidas pelos agentes que a habitam.

Agent (Agente): agentes do sistema. Os agentes podem habitar ambientes e organizações, onde por sua vez são membros de sua cultura e devem desempenhar papéis. Agentes possuem objetivos, ações, planos, crenças, valores e tem seu comportamento regulado por normas.

Role (Papel): papéis que agentes e (sub)organizações podem desempenhar. Os papéis podem possuir crenças, ações, objetivos, normas que restringem o comportamento do papel.

Belief (Crença): crença/conhecimento que o agente possui sobre o sistema, os agentes do sistema e ele mesmo.

Goal (Objetivo): objetivos que devem ser alcançados pelos agentes do sistema.

Action (Ação): ações executadas pelos agentes do sistema. Uma ação pode promover e rebaixar valores do agente quando é executada.

Plan (Plano): plano que define um conjunto de ações para alcançar um objetivo.

Culture (Cultura): cultura construída pelas entidades de uma organização. Os agentes são membros das culturas das organizações que habitam.

Value (Valor): valor pessoal de um agente que o auxilia na seleção ou avaliação de comportamentos ou eventos.

Norm (Norma): crença que regula ações dos agentes. As normas definem as ações que podem ser executadas pelos agentes (permissões), as ações que devem ser executadas pelos agentes (obrigações) e as ações que não devem ser executadas pelos agentes (proibições). Uma norma que está associada a um papel, se aplica a todos os agentes que desempenham este papel; uma norma que está associada a uma organização ou a um ambiente se aplica a todos os agentes que habitam aquela organização ou ambiente.

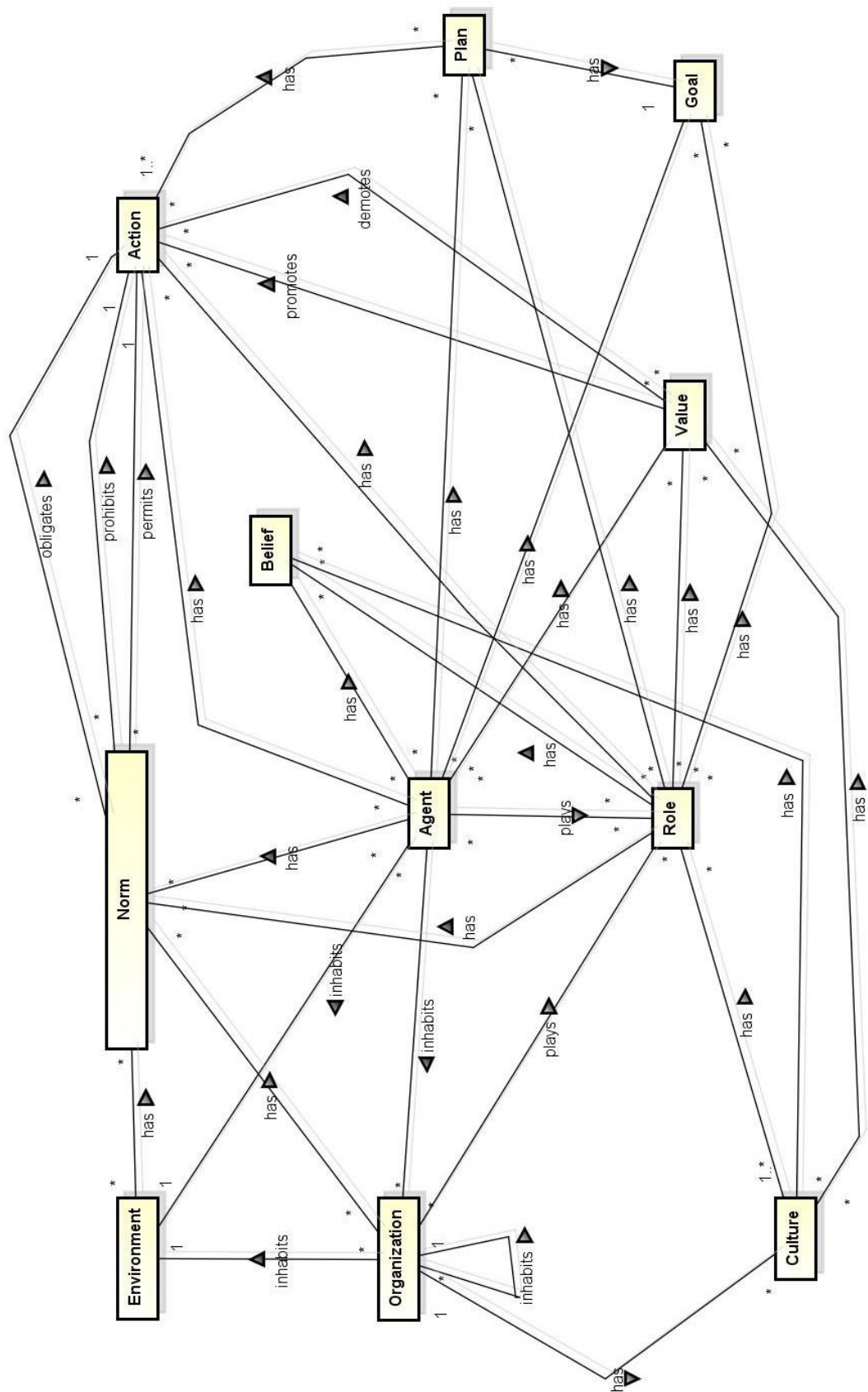


Figura 2 - Ontologia de SMA Culturais
Fonte: Marques e Figueiredo, 2014

1.3. Modelagem, Modelo e Metamodelo

A modelagem, i.e. a ação de criar modelos, é uma das atividades mais citadas a serem realizadas nas etapas de Análise e Projeto de Software por Pressman (1992) e Sommerville (2003). A finalidade de uma modelagem é manter a organização e o cumprimento dos requisitos na etapa de desenvolvimento. Os modelos auxiliam a ter uma visão mais abrangente do funcionamento do sistema.

Criar um modelo é o processo de converter uma visão da realidade em uma representação dela (VAN GIGCH, 1991). Modelo é a forma simplificada de abstrair fenômenos do mundo real. E da mesma forma que abstraímos para criar um modelo é possível aplicar a mesma abstração para a modelagem em si. O ato de abstrair um modelo, criando um modelo da modelagem é chamado de metamodelo (VAN GIGCH, 1991). A Figura 3 demonstra um exemplo das atividades de metamodelagem, modelagem e instanciação de um modelo.

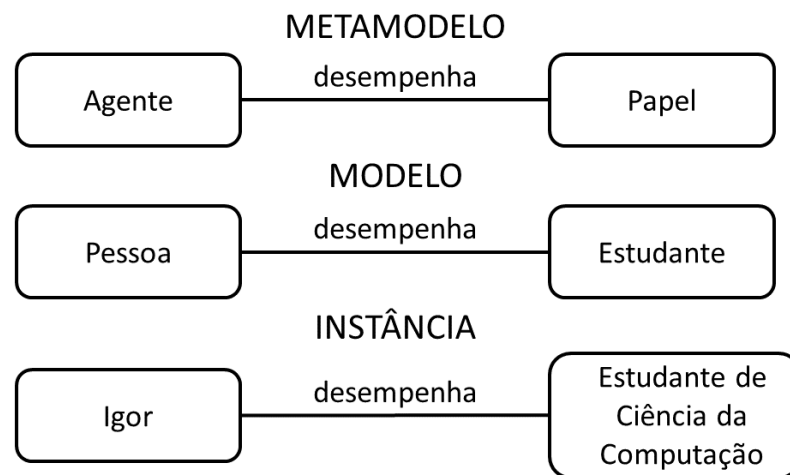


Figura 3 - Exemplo de metamodelo

Assim, o metamodelo pode ser visto como uma ferramenta para representar um conjunto de conceitos de um determinado domínio para a modelagem dos mesmos. A ontologia para SMA Culturais de Marques e Figueiredo (2014) foi utilizada como metamodelo para a elaboração da ferramenta de modelagem proposta nesse trabalho.

2. MATERIAIS, TÉCNICAS E MÉTODOS

2.1. A Ferramenta e Ambiente Eclipse

O Eclipse é um Ambiente de Desenvolvimento Integrado (IDE) que foi desenvolvido em Java e segue o modelo de software de código aberto. O ambiente suporta o desenvolvimento de softwares utilizando diversas linguagens tais como C/C++, PHP, ColdFusion, Python, Scala e plataforma Android, sendo que para isso é necessária a utilização de plug-ins que são disponibilizados pela própria Fundação Eclipse ou pela comunidade de usuários desta IDE. O grande número de plug-ins fornecidos pelo Eclipse tem o objetivo atender as diferentes necessidades de diferentes programadores, mas a sua principal e mais famosa utilização é para o desenvolvimento de softwares na linguagem Java (ECLIPSE FOUNDATION, 2014).

O Eclipse foi escolhido como IDE a ser utilizada no desenvolvimento de nossa ferramenta por alguns motivos, tais como: o suporte à linguagem Java, a grande quantidade de plug-ins, em especial a existência do plug-in GMF, o grande número de usuários desenvolvedores, permitir a unificação entre modelagem e desenvolvimento para o especialista de agentes e SMA, e por contar com uma grande comunidade para auxiliar quando problemas são encontrados. A versão do Eclipse utilizada neste trabalho é a Kepler Service Release 2.

O Eclipse (Figura 4) possui uma barra de menu localizado superiormente contendo todas as opções de criação, edição, navegação, pesquisa, execução, atualização e ajuda, abaixo da barra de menu há uma caixa de ferramenta com ícones das opções que geralmente o usuário mais utiliza, essa caixa de ferramenta facilita o acesso rápido as funcionalidades oferecidas pela IDE, tais como: criação e salvamento de arquivos, execução e inspeção de código, criação de classes e pacotes, entre outras.

No canto esquerdo do Eclipse há uma área reservada para a exploração de pacotes, possibilitando a navegação pelos projetos, pastas e pacotes criados pelo usuário. A área central do Eclipse é reservada para exibição dos arquivos a serem abertos pelo usuário, o Eclipse possibilita que o usuário abra mais de um arquivo e o mesmo aberto gerenciando sua exibição em abas.

Ainda na área central do Eclipse há um espaço reservado localizado abaixo a área de exibição de arquivos, esta área permite que o usuário visualize os problemas

que podem ocorrer ao executar seu código, as mensagens transmitidas pelo console e também permite o usuário alterar as propriedades de determinados tipos de arquivos, em alguns casos é possível ter opções extras referente ao projeto criado, e. g., o GMF *dashboard*. Na área localizada mais à direita do Eclipse fica a *Task List* (lista de tarefas) e o *Outline* (janela de esboço).

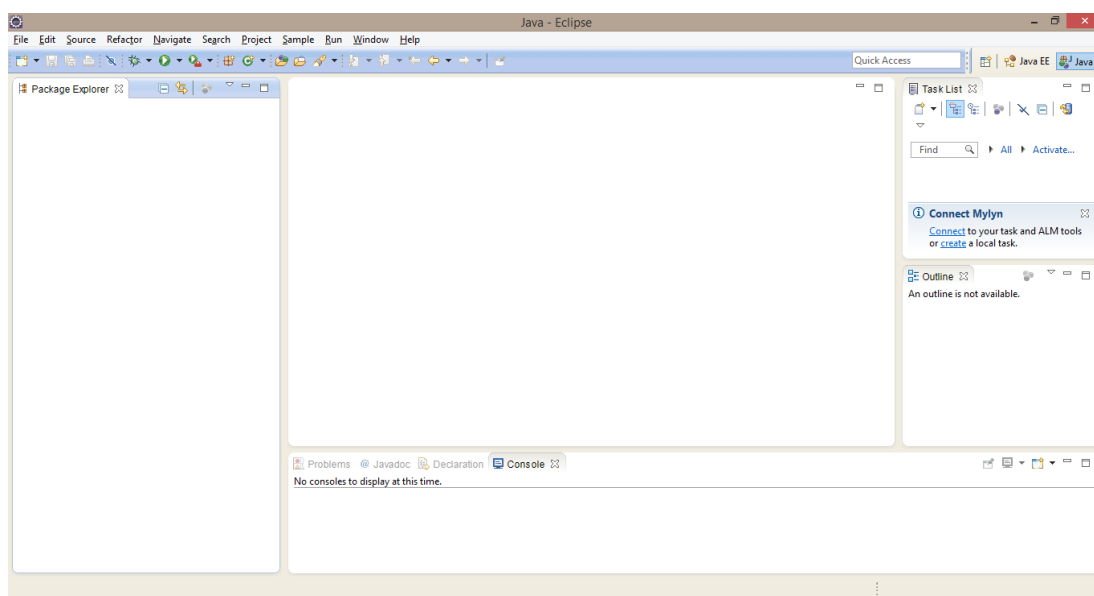


Figura 4 - Tela principal da IDE Eclipse

2.2. Linguagem Java

Java é uma linguagem de programação orientada a objetos desenvolvida na década de 90 por uma equipe de programadores chefiada por James Gosling, na empresa Sun Microsystems. Seu principal objetivo era resolver alguns problemas encontrados em outras linguagens naquela década, tais como gerenciamento de ponteiros, gerenciamento de memória, organização no código, falta de bibliotecas e ter de reescrever parte do código ao mudar de sistema operacional (CAELUM, 2014).

Diferentemente das linguagens convencionais, que são compiladas para código nativo, a linguagem Java é compilada para um *bytecode* que é executado por uma máquina virtual.

A linguagem Java é uma das linguagens mais utilizadas atualmente e sua grande gama de recursos e bibliotecas foram relevantes em sua escolha para o desenvolvimento da ferramenta. Outra vantagem de ter escolhido a linguagem Java é o fato dela ser multiplataforma, assim não depende em qual sistema operacional se

utiliza. Com a escolha da linguagem Java isso possibilitou o uso do framework GMF que também é fundamental para o desenvolvimento da ferramenta.

2.3. Java Development Kit (JDK)

Para criar *applets* e aplicações Java, são necessárias ferramentas de desenvolvimento como o JDK. O JDK é o Kit de Desenvolvimento Java que inclui o *Java Runtime Environment*, o compilador Java e as APIs Java (ORACLE, 2014).

Java Runtime Environment significa Ambiente de Tempo de Execução Java e é utilizado para executar as aplicações da plataforma Java (CAELUM, 2014).

O JDK está disponível para diversos Sistemas Operacionais, dentre eles estão o Windows, Linux, Mac OS X e Solaris SPARC. Esta diversidade permite o desenvolvimento de softwares em Java independente de qual SO está sendo utilizado, desta forma quando se fizer necessário a troca do SO, não há necessidade de reescrita do código desenvolvido (ORACLE, 2014).

2.4. Graphical Modeling Framework (GMF)

O GMF é um framework para desenvolvimento de editores gráficos para modelos de domínio dentro da plataforma Eclipse (Figura 5). Ele foi baseado em outros dois frameworks denominados GEF (*Graphical Editing Framework*), utilizado para a criação de editores gráficos genéricos e EMF (*Eclipse Modeling Framework*), que permite ao desenvolvedor construir metamodelos e gerar códigos Java referidos ao mesmo (ECLIPSE FOUNDATION, 2014).

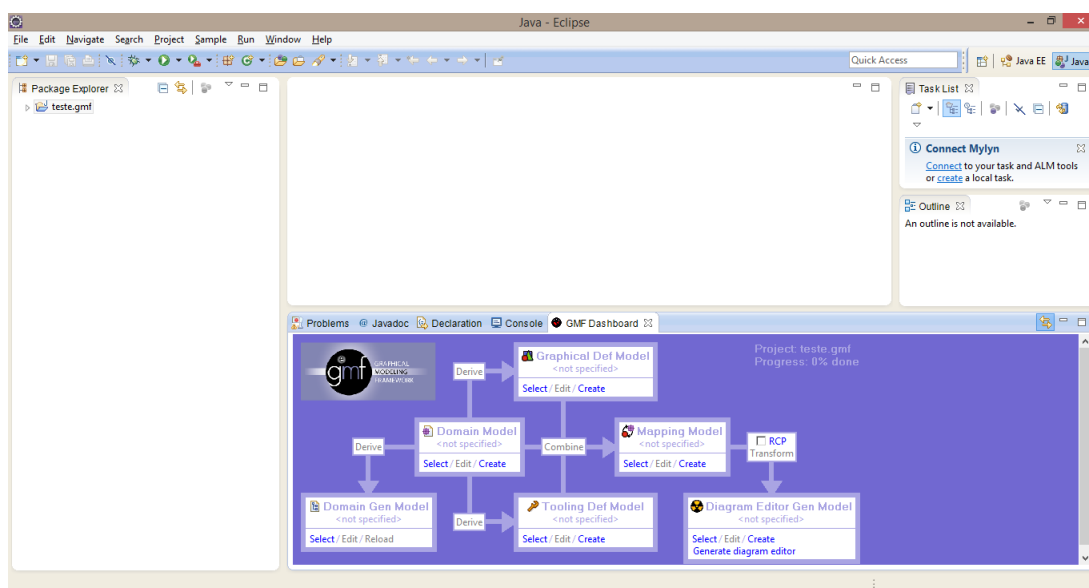


Figura 5 - Tela GMF dashboard no Eclipse

A utilização do GMF neste trabalho é de suma importância para o desenvolvimento da ferramenta, a partir do modelo de domínio criado no framework é possível chegar a conclusão da ferramenta, visualizando o diagrama do metamodelo, obtendo paletas de criação e gerando o código em Java de toda ferramenta.

2.5. Desenvolvimento

Para iniciar o desenvolvimento da ferramenta é necessário possuir a IDE Eclipse instalada. Foi utilizada a versão mais atual disponível (Kepler Service Release 2). Fez-se necessário também ter instalado e configurado no sistema operacional o JDK em sua versão atual (jdk 1.8.0_05) e ter o plug-in GMF instalado no Eclipse.

2.5.1. Criando o projeto GMF

Para criar um projeto GMF, basta estar com o Eclipse aberto e em sua parte superior acessar *File > New > Other* (Figura 6), em seguida é exibida uma janela com todas as opções de projetos possíveis de serem criados nesta IDE. Então, seleciona-se a pasta *Graphical Modeling Framework* para expandir, e seleciona-se pôr fim a opção *Graphical Editor Project* (Figura 7).

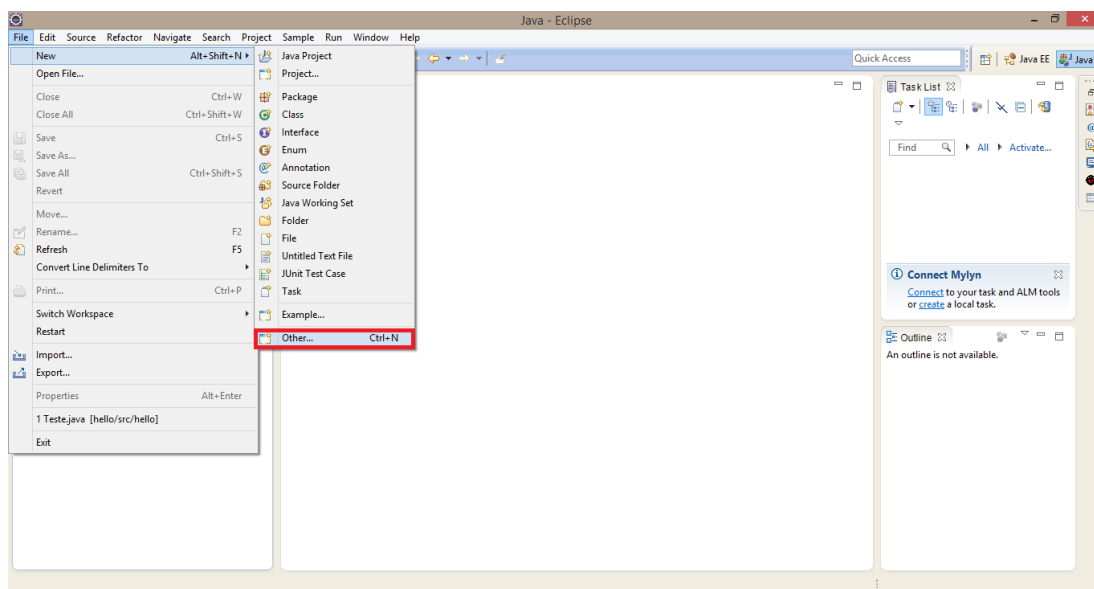


Figura 6 - Tela de como criar um projeto GMF no Eclipse

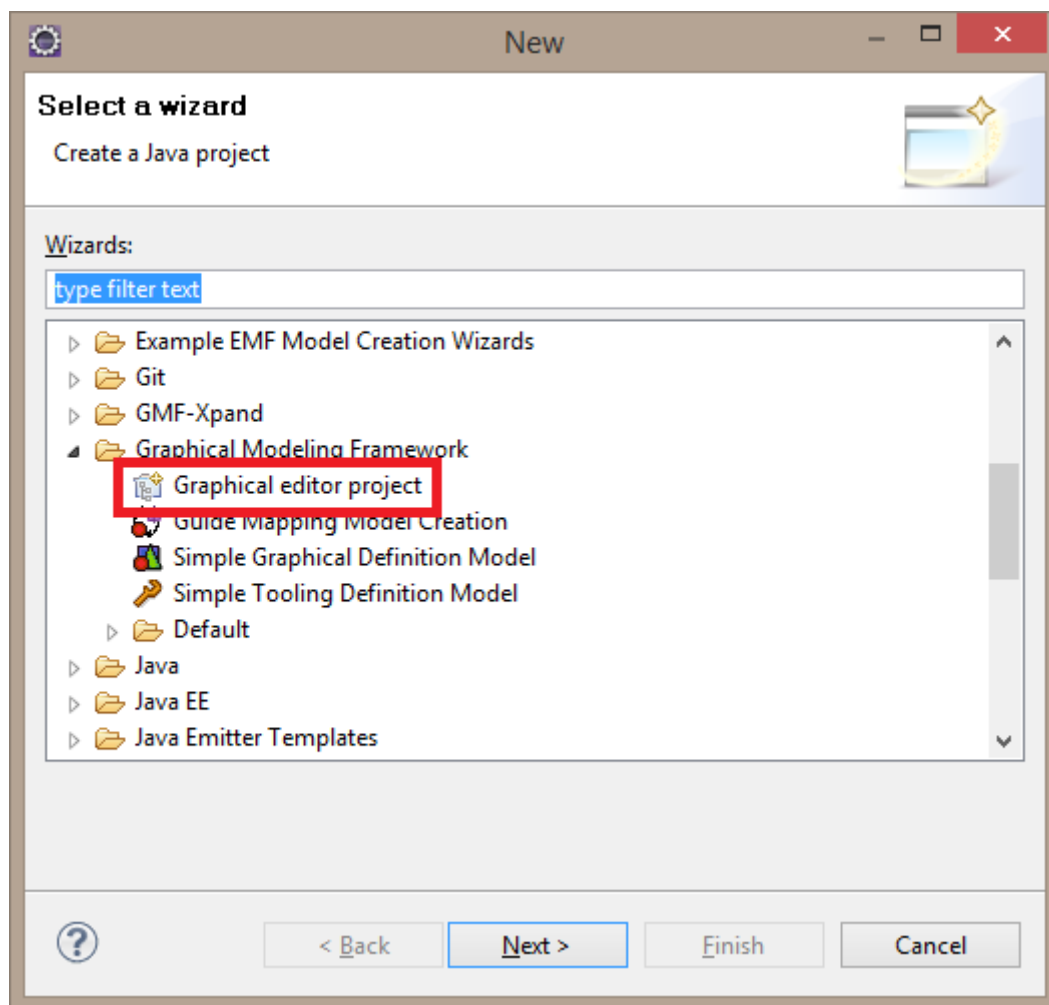


Figura 7 - Tela de seleção do projeto GMF

Uma nova janela é aberta, então é necessário colocar o nome do projeto, por exemplo teste.gmf, e selecionar o local onde ele será salvo ou apenas deixar no local padrão. Outra janela será exibida, nesta janela é marcada a opção *Show dashboard view for the created Project*, essa opção torna visível a *dashboard* do GMF no projeto, para finalizar a criação do projeto basta clicar no botão *Finish* (Figura 8).

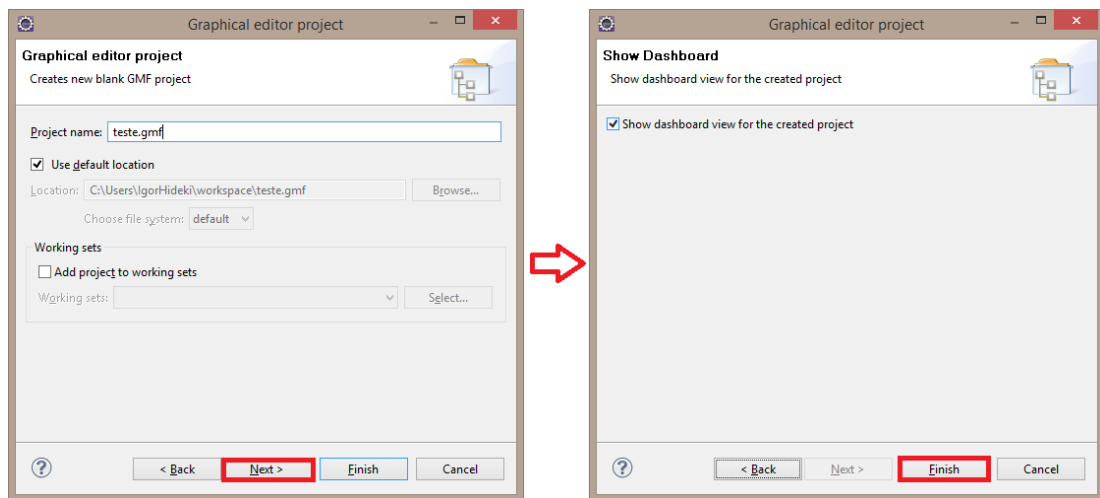


Figura 8 - Telas de definição de nome do projeto GMF e exibição da GMF dashboard

2.5.2. Criando o metamodelo

A criação do metamodelo baseia-se em regras, e as regras que o definem neste projeto são as da ontologia de SMA Culturais mostradas anteriormente na Seção 1.2.2.

Para criar o metamodelo é necessário a criação de um modelo de domínio cuja a extensão do arquivo é .ecode. Dentro do *dashboard* do GMF há um retângulo intitulado *Domain Model* com as opções de *Select / Edit / Create*, como neste projeto está sendo criado todos os elementos da ferramenta do início é necessário a criação do modelo de domínio clicando em *Create* (Figura 9). Com o modelo de domínio criado é possível configurar suas propriedades (Figura 10), posteriormente são configuradas as classes (Figura 11 e Figura 12), seus atributos e referências de acordo com as regras definidas anteriormente.

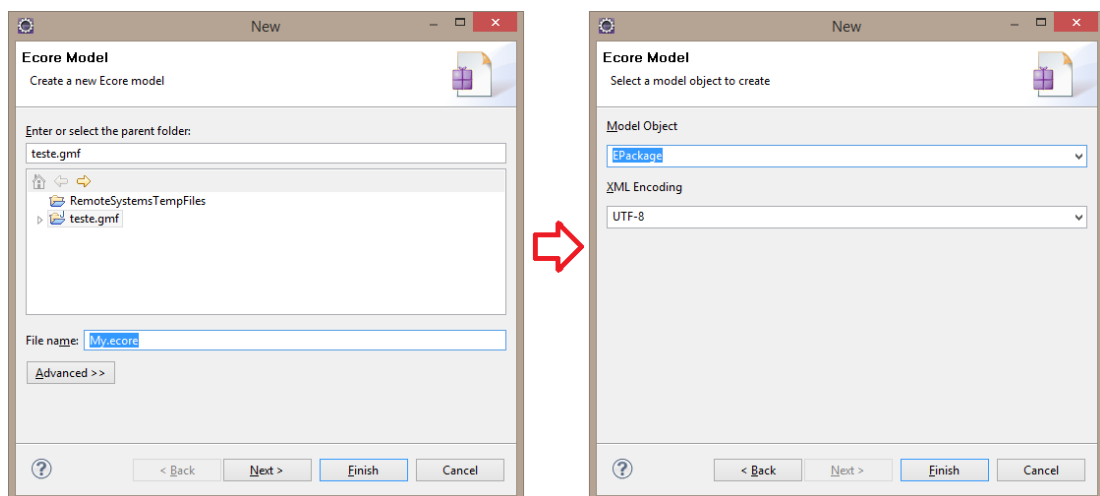


Figura 9 - Tela de criação do Modelo de Domínio

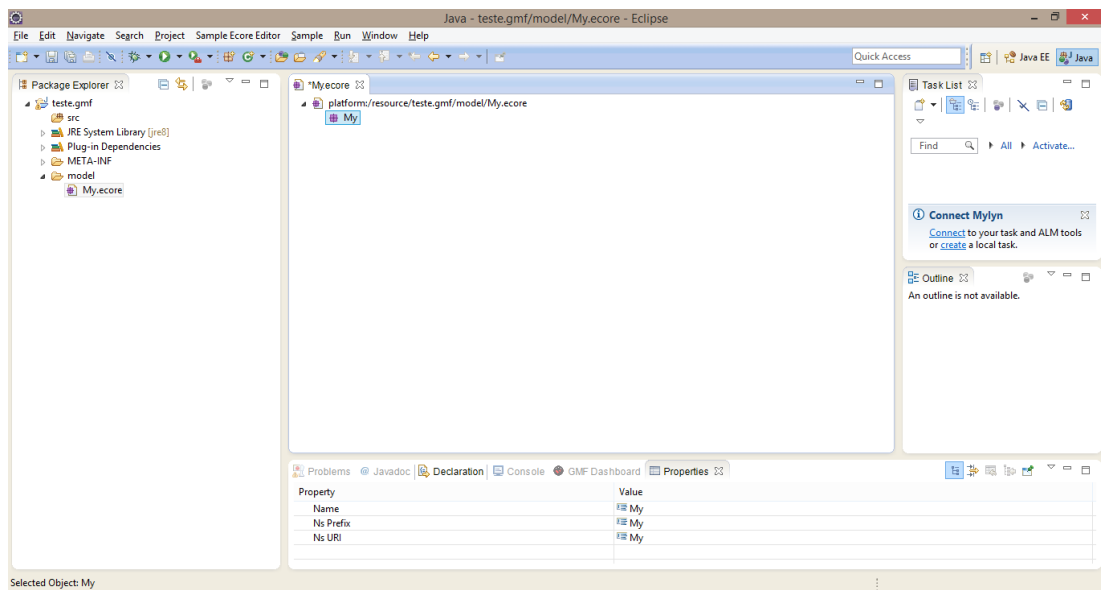


Figura 10 - Tela de definição das propriedades do Modelo de Domínio

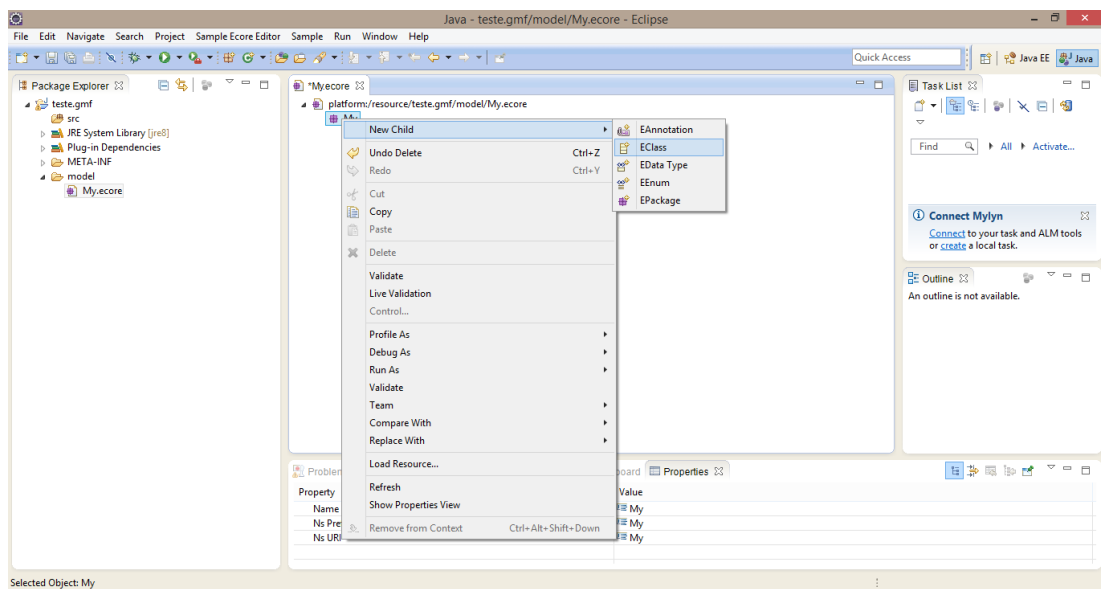


Figura 11 - Tela de como criação de classes no Modelo de Domínio

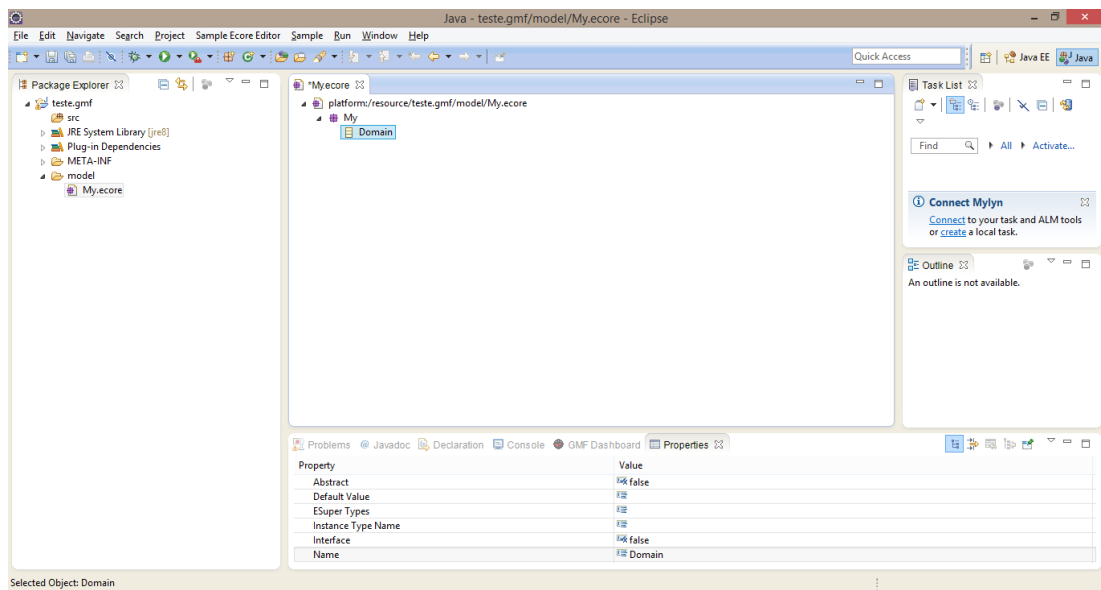


Figura 12 - Tela de definição das propriedades das classes no Modelo de Domínio

As classes, seus atributos e referências podem ser configurada na aba de propriedades do Eclipse (Figura 13, Figura 14 e Figura 15 respectivamente). No metamodelo as classes devem estar contidas em um domínio, por isso é criado uma classe domínio onde todas as outras classes são referenciadas.

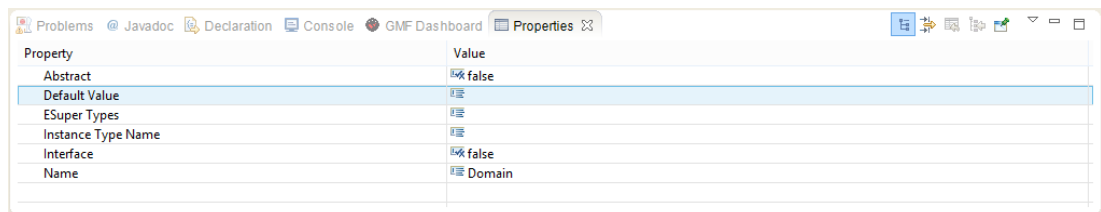


Figura 13 - Tela da aba de propriedades das classes de um Modelo de Domínio

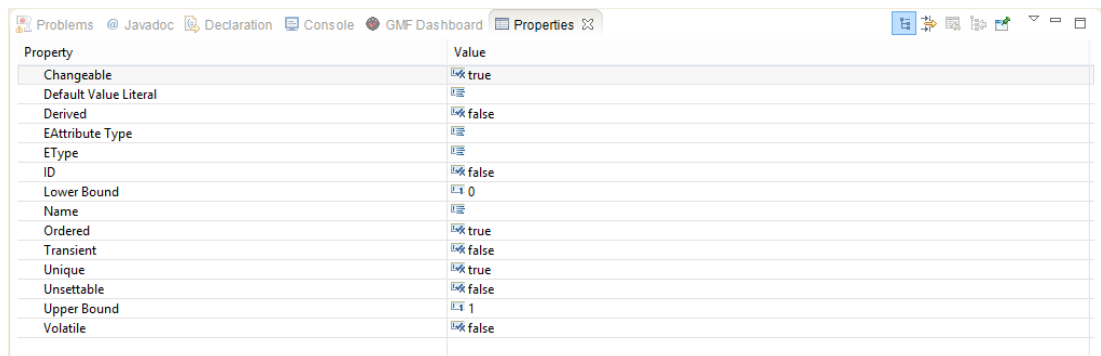


Figura 14 - Tela da aba de propriedades dos atributos de um Modelo de Domínio

Property	Value
Changeable	true
Container	false
Containment	false
Default Value Literal	
Derived	false
EKeys	
EOpposite	
EType	
Lower Bound	0
Name	
Ordered	true
Resolve Proxies	true
Transient	false
Unique	true
Unsettable	false
Upper Bound	1
Volatile	false

Figura 15 - Tela da aba de propriedades dos relacionamentos de um Modelo de Domínio

Após a criação das classes é gerado o diagrama ecore de acordo com o modelo criado, para gerar este diagrama basta clicar com o botão direito no arquivo do modelo e selecionar a opção *Inicialize ecore_diagram diagram file* (Figura 16), logo após é aberta uma janela que permite escolher onde deseja salvar o diagrama.

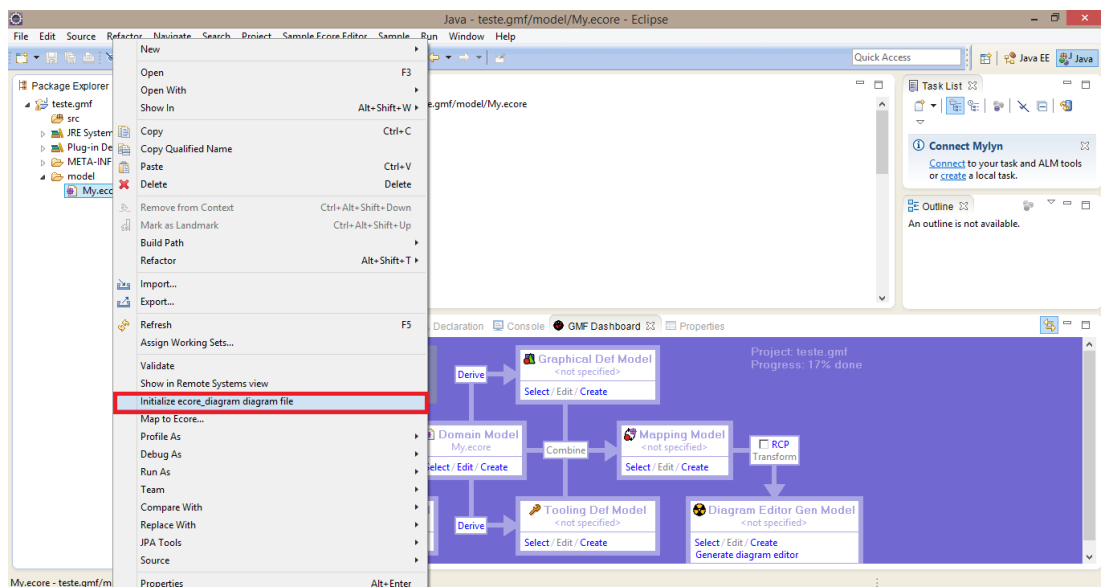


Figura 16 - Tela de como gerar o diagrama referente ao Modelo de Domínio

Com o modelo de domínio e o diagrama criados é possível criar o modelo de geração dos códigos das classes em Java. Para criar o modelo de geração dos códigos das classes basta ir no *dashboard* do GMF, clicar no retângulo pequeno intitulado *Derive* que se encontra na seta que aponta para o retângulo maior intitulado *Domain Gen Model* (Figura 17). Uma nova janela aparecerá, nela é possível escolher onde o modelo será salvo.

Nas telas seguintes é preciso indicar onde está salvo o modelo de domínio a ser utilizado para a geração dos códigos das classes, depois de seguir esses passos é criado o modelo. Para gerar os códigos em Java basta ir no *gen model* criado e clicar com o botão direito no modelo que aparece, quando clicar com o botão direito aparecerão as opções *Generate Model Code* e *Generate Edit Code*, selecione uma de cada vez na ordem citada e o código das classes é gerado (Figura 18).

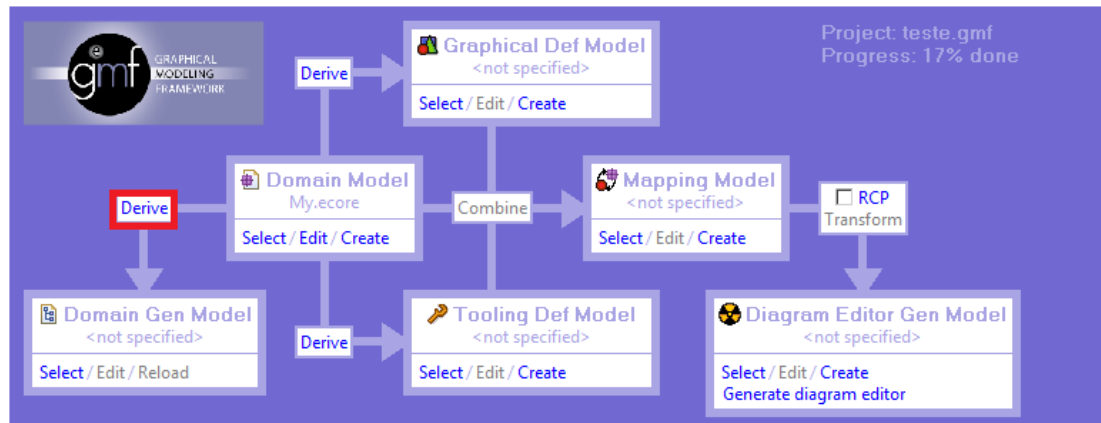


Figura 17 - Tela de como criar um Modelo de Geração de Código no dashboard do GMF

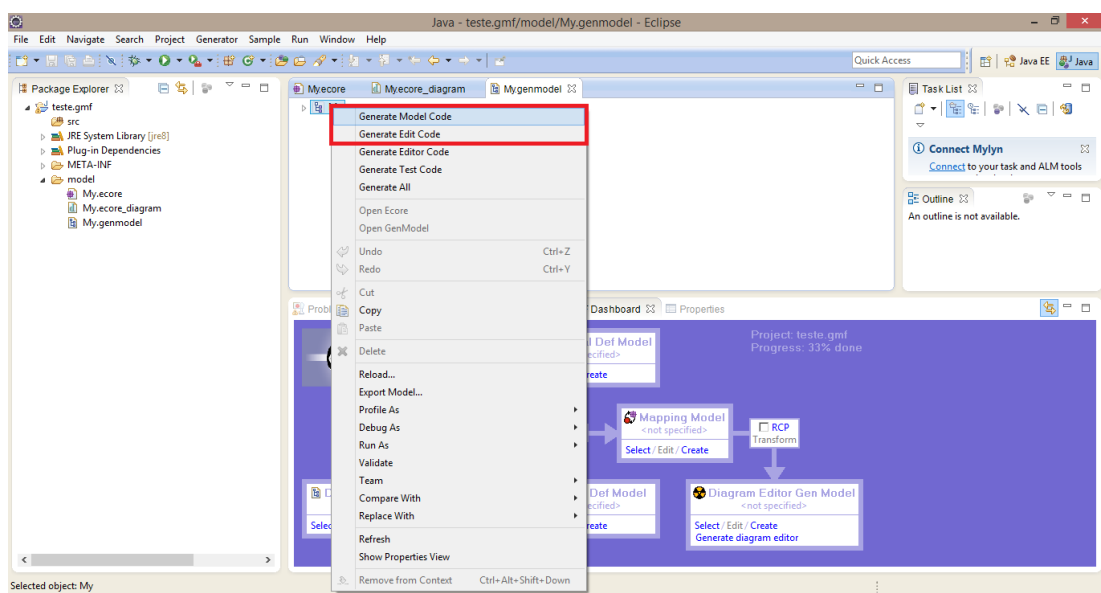


Figura 18 - Tela de como gerar os códigos das classes no Modelo de Geração de Código

2.5.3. Criando modelo gráfico

O modelo gráfico é o modelo que define as formas que as classes e atributos assumirão graficamente na ferramenta final. No *dashboard* do GMF há uma seta saindo do retângulo intitulado *Domain Model* que tem como alvo o retângulo *Graphical Def Model*, na seta tem o botão *Derive* como no caso anterior, para gerar a

classe é necessário clicar no botão e uma janela de configurações será aberta, também como em casos anteriores é possível escolher o nome do modelo a ser gerado, na próxima janela é necessário a escolha da classe que será o domínio que todas as outras classes pertencem, em seguida é possível configurar as classes que serão representadas graficamente por figuras e as classes que terão arestas de ligação (Figura 19).

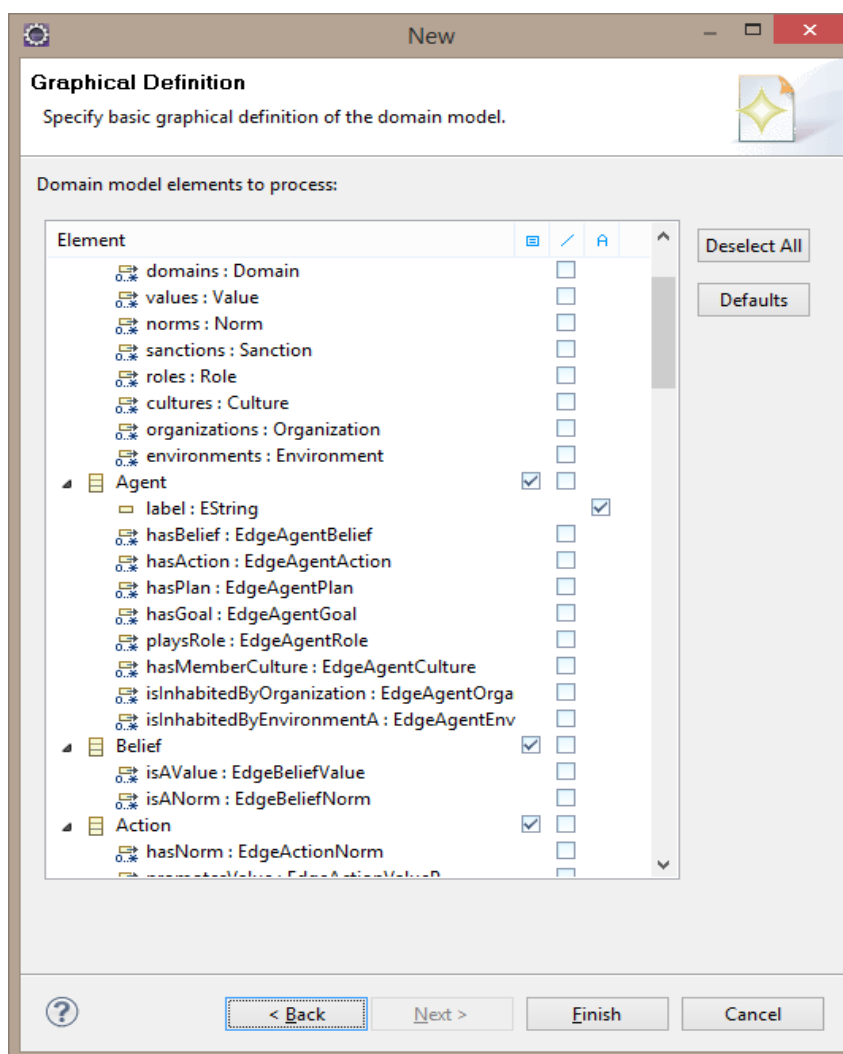


Figura 19 - Tela de definição gráfica das classes

Com o modelo gráfico já gerado é possível visualizar quais classes se tornaram nós e quais são apenas classes de conexão. Neste modelo, alterações individuais podem ser feitas como distinguir e alterar as figuras de cada classe e seus rótulos (Figura 20).

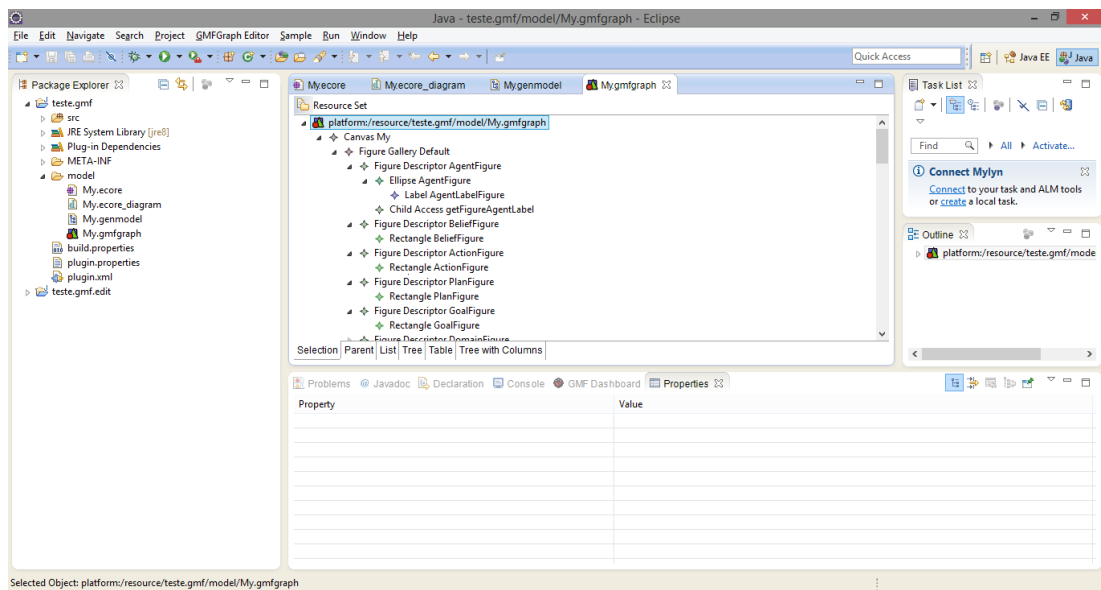
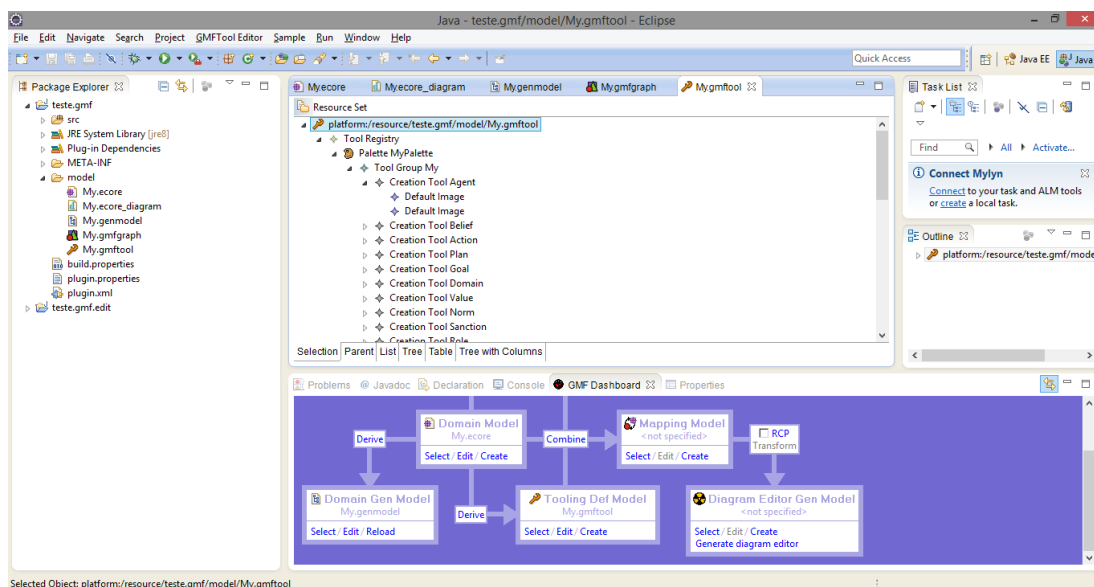
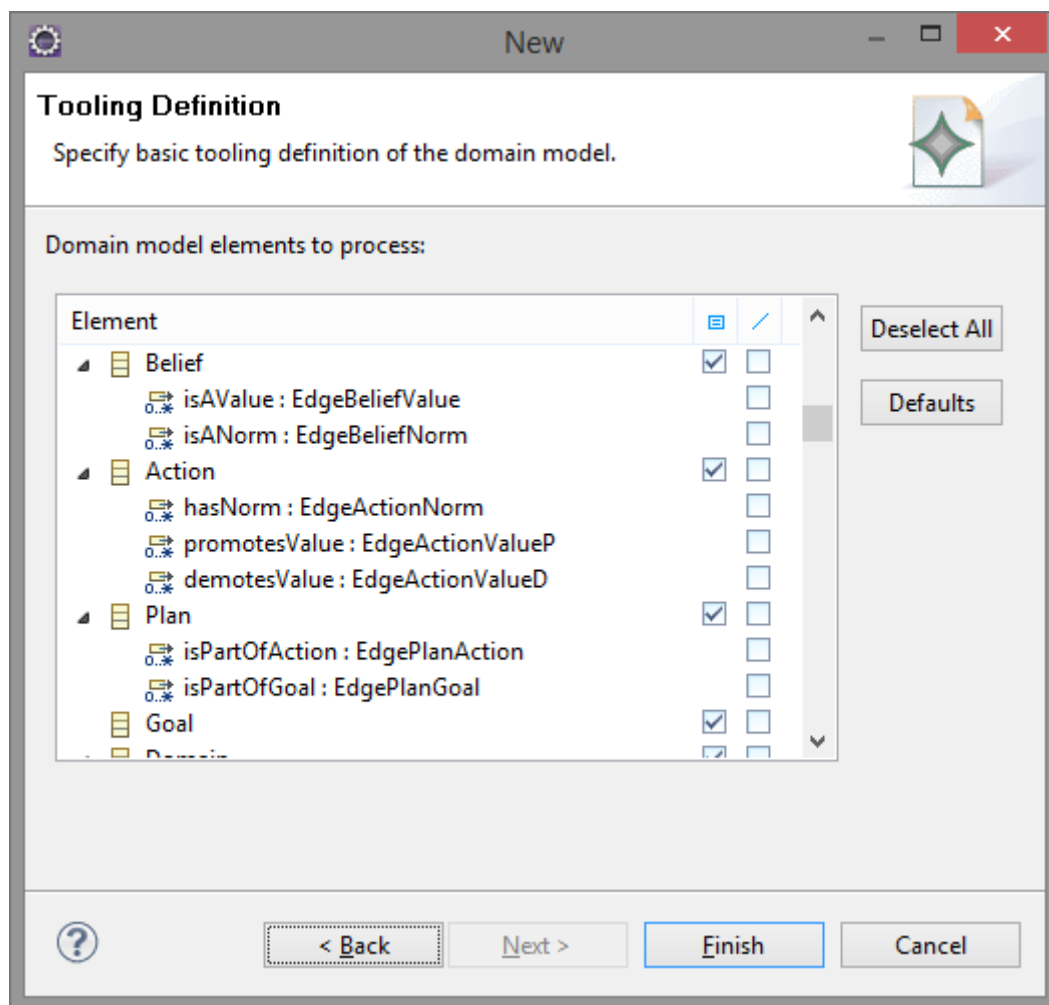


Figura 20 - Tela de configuração do Modelo Gráfico

2.5.4. Criando a paleta gráfica

A paleta gráfica é a caixa de ferramenta da aplicação, ela é a parte da ferramenta que permite a interação do usuário com seu diagrama, pela paleta o usuário seleciona o que deseja criar em seu modelo. Para gerar o modelo de definição da paleta é necessário ir no *dashboard* do GMF e clicar no botão *Derive* situado na seta saindo do *Domain Model* com destino ao *Tooling Def Model*. Como na criação dos outros modelos, será exibido uma nova janela possibilitando escolher a localização para o salvamento do modelo de definição da paleta gráfica, em seguida é selecionado a classe domínio do metamodelo e as figuras e ligações que serão exibidas na paleta gráfica (Figura 21). Após a criação do modelo de definição da paleta é possível fazer alterações individuais caso haja a necessidade (Figura 22).



2.5.5. Criando modelo de mapeamento

O modelo de mapeamento é um conjunto de referências dos outros modelos criados anteriormente, nele são feitas as associações das classes com os componentes gráficos e com os elementos da paleta. Este modelo é responsável por gerir o mapeamento do diagrama e para cria-lo basta clicar no botão Combine localizado no *dashboard* do GMF, uma sequência de janelas será aberta com algumas janelas semelhantes aos modelos criados anteriormente. As janelas novas que surgirem são para selecionar o modelo de domínio, o modelo gráfico e o modelo da paleta, após fazer as seleções é necessário verificar se todos os componentes associados às classes (nós e ligações) estão corretos (Figura 23). Após a criação do modelo de mapeamento é possível fazer verificações e alterações individuais nos elementos que assim o compõem (Figura 24).

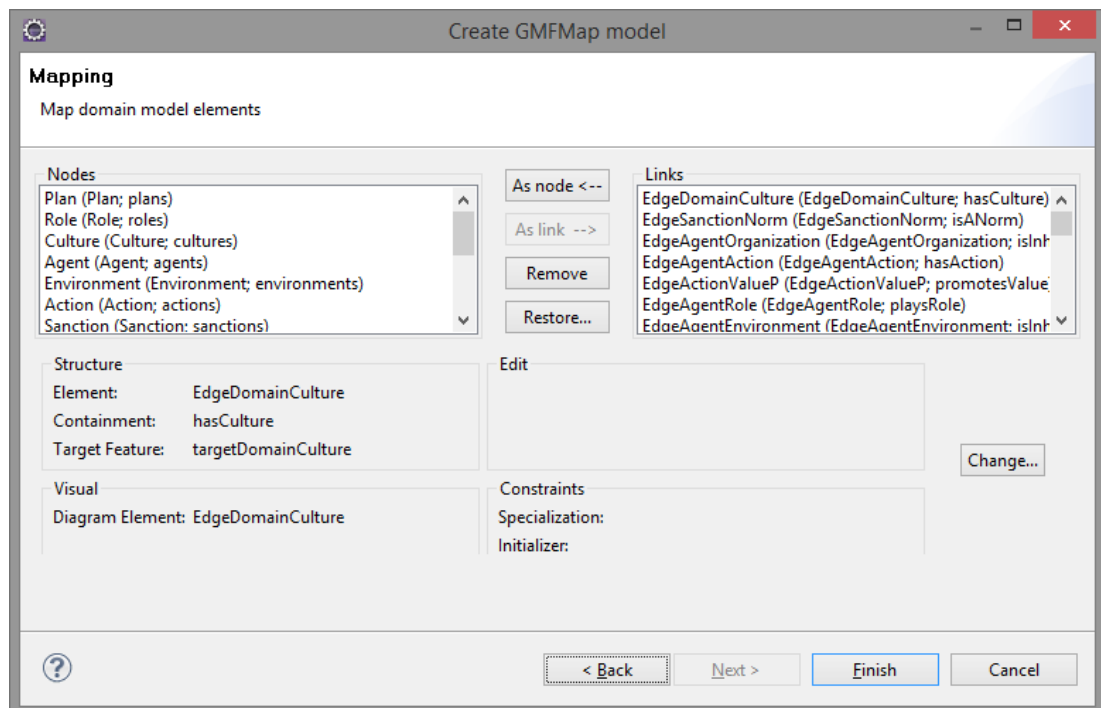


Figura 23 - Tela de configuração e associação dos elementos as classes

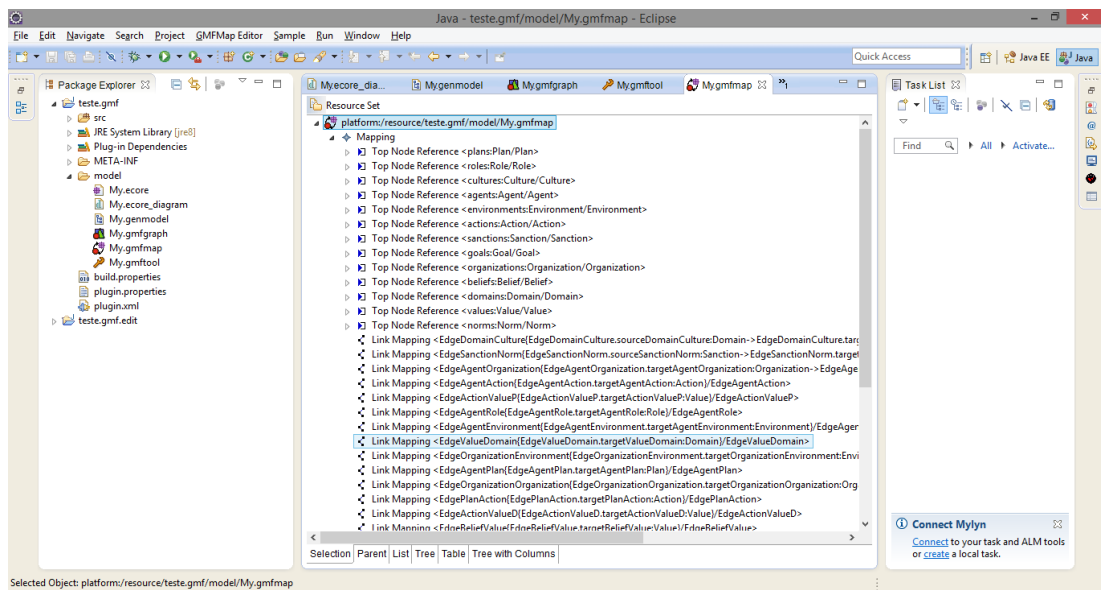


Figura 24 - Tela de configuração do Modelo de Mapeamento

2.5.6. Criando modelo editor de diagramas e gerando o projeto executável

O modelo editor de diagramas é o último modelo necessário para a conclusão da ferramenta, para criá-lo o processo é semelhante aos anteriores, no *dashboard* do GMF tem o botão intitulado *Transform* clicando nele é gerado modelo editor de diagramas (Figura 25). Ainda no *dashboard* dentro retângulo intitulado *Diagram Editor Gen Model* clique em *Generate diagram editor* (Figura 26), assim o projeto executável da ferramenta será gerado e ela estará finalizada, para utilizar a ferramenta basta executar o projeto.

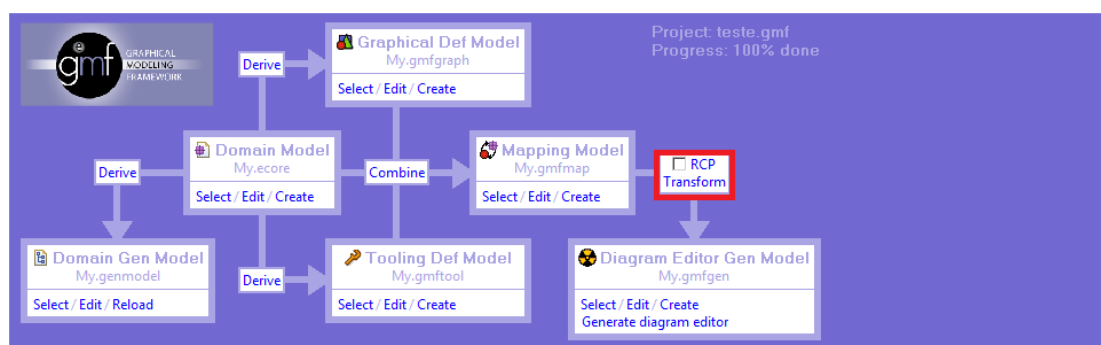


Figura 25 - Tela de criação do Modelo Editor de Diagramas no GMF *dashboard*

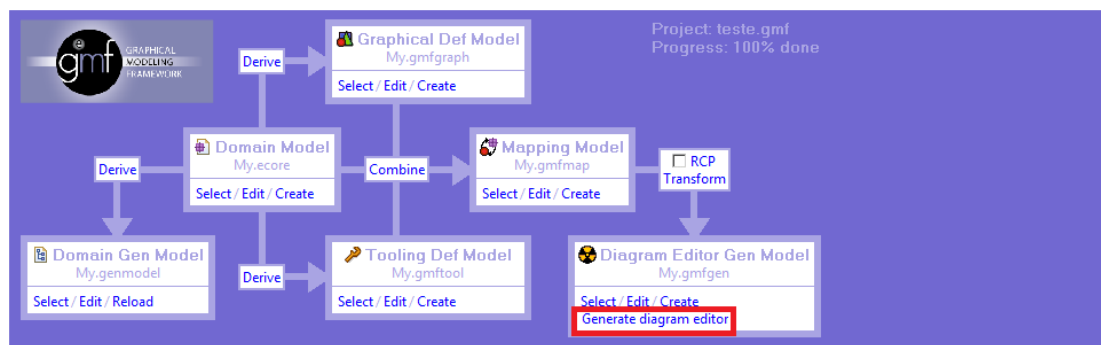


Figura 26 - Tela da geração do projeto executável da ferramenta

3. RESULTADOS

O resultado obtido com a realização deste trabalho é a ferramenta de modelagem de SMA Culturais concluída. Através dessa ferramenta, analistas de sistemas orientados a agentes podem construir a modelagem das entidades do SMA e seus relacionamentos, considerando os aspectos culturais desses sistemas, e.g. normas, valores, papéis, etc.

Para o desenvolvimento da ferramenta foram utilizados os materiais, técnicas e métodos, tais como: IDE Eclipse, Linguagem de programação Java, *Java Development Kit* (JDK) e *Graphical Modeling Framework* (GMF) descritos no capítulo anterior.

A ferramenta funciona como um plug-in para o Eclipse a sua aparência final é apresentada na Figura 27.

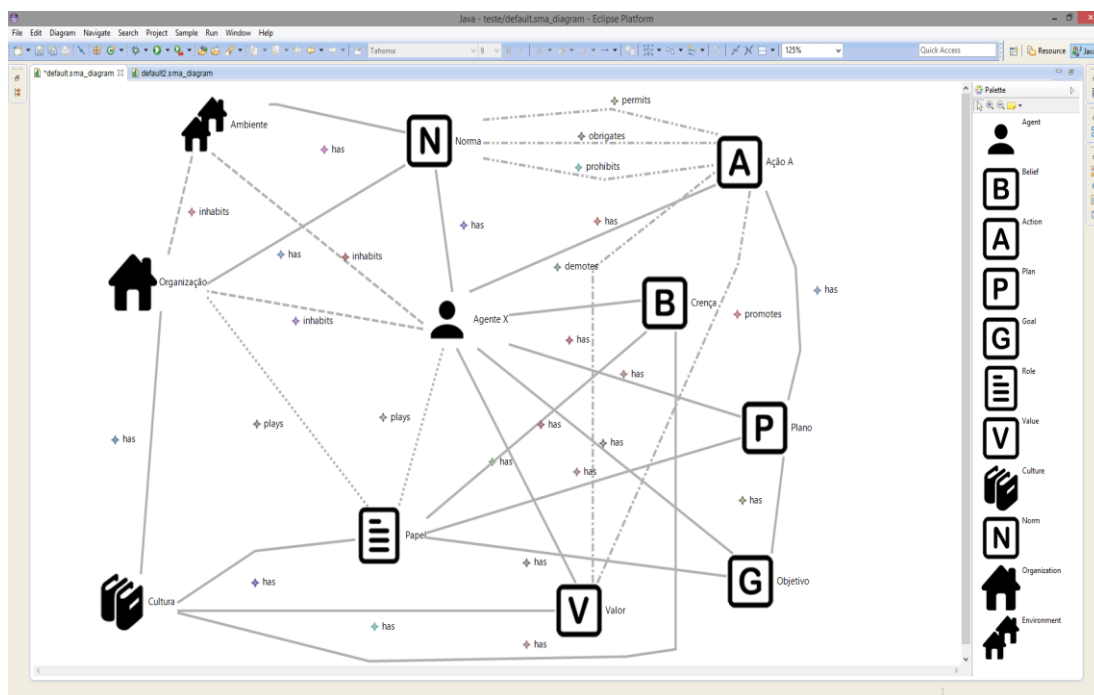









Figura 27 - Exemplo de modelagem na ferramenta obtida neste trabalho

O modelo de domínio utilizado para a criação da ferramenta foi baseado na ontologia apresentada por Marques e Figueiredo (2014), respeitando todas as classes, relacionamentos e cardinalidades. Desta forma, quando o analista tenta criar um

relacionamento que não é possível (considerando as regras da ontologia) entre duas entidades, a ferramenta apresenta uma mensagem de erro.

Já o modelo gráfico foi gerado de acordo com o modelo de domínio e fez-se necessário a alteração das figuras padrão utilizadas para representação de elementos na ferramenta. Sua alteração tem o objetivo de tornar as figuras mais intuitivas para o usuário final, também foi alterado as representações de arestas para cada tipo de relacionamento e adicionado rótulos para a intitulação delas. Assim, as Tabelas 1 e 2 apresentam respectivamente os símbolos que representam cada uma das entidades e os relacionamentos que podem ser utilizados nos modelos de SMA Culturais.

Tabela 1 - Entidades dos modelos de SMA Culturais

	<i>Action (Ação)</i>
	<i>Agent (Agente)</i>
	<i>Belief (Crença)</i>
	<i>Culture (Cultura)</i>
	<i>Environment (Ambiente)</i>
	<i>Goal (Objetivo)</i>
	<i>Norm (Norma)</i>
	<i>Organization (Organização)</i>










	<i>Plan (Plano)</i>
	<i>Role (Papel)</i>
	<i>Value (Valor)</i>

Tabela 2 - Relacionamentos dos modelos de SMA Culturias

	<i>Has (Tem)</i>
	<i>Inhabits (Habita)</i>
	<i>Plays (Desempenha)</i>
	<i>Demotes (Rebaixa)</i>
	<i>Promotes (Promove)</i>

 obligates 	<i>Obligates (Obriga)</i>
 permits 	<i>Permits (Permite)</i>
 prohibits 	<i>Prohibits (Proíbe)</i>

Para facilitar a distinção dos elementos e auxiliar o usuário foram adicionados rótulos em que o usuário pode nomear os elementos de sua modelagem (Figura 28).

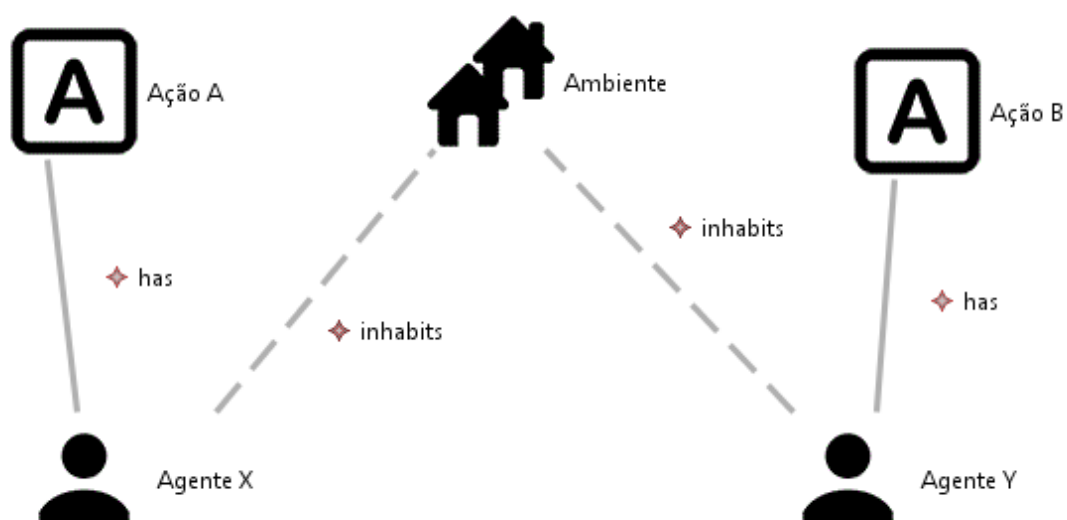


Figura 28 - Exemplo dos elementos, arestas e rótulos

A paleta da ferramenta no Eclipse contém os ícones representando cada elemento, seguido pelo seu nome para facilitar o uso da ferramenta de modelagem de SMA Culturais pelo usuário (Figura 29).

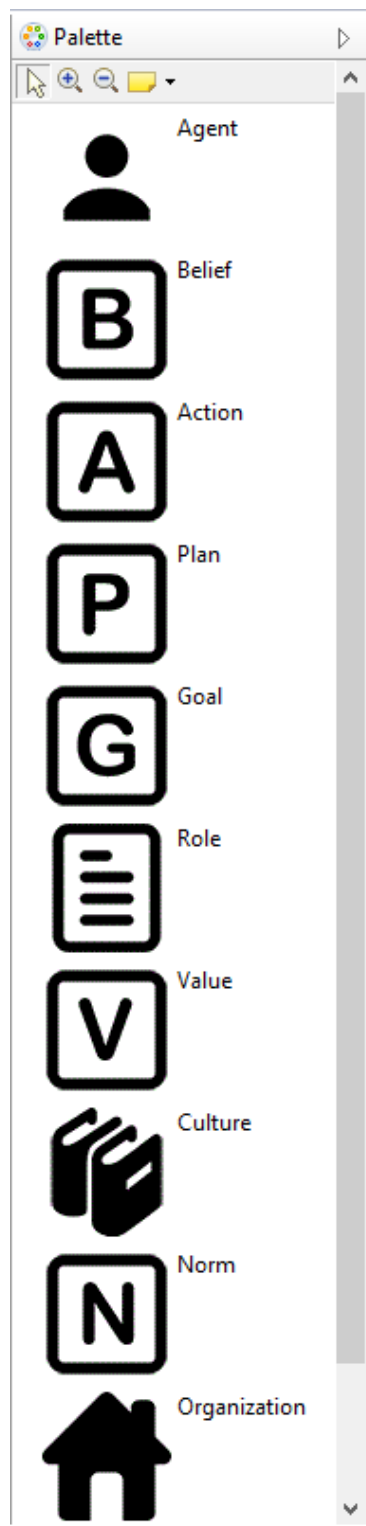


Figura 29 - Paleta da ferramenta de modelagem de SMA Culturais

A Figura 30 demonstra um exemplo de uma modelagem de um SMA Cultural desenvolvido na ferramenta obtida como resultado deste trabalho. Nesse exemplo, podemos observar o agente Igor que desempenhando o papel *Computer Science*

Student deve seguir a norma N1 que o obriga a executar a ação *PresentInternshipReport* na organização *Computer Science Department* do ambiente UFMT.

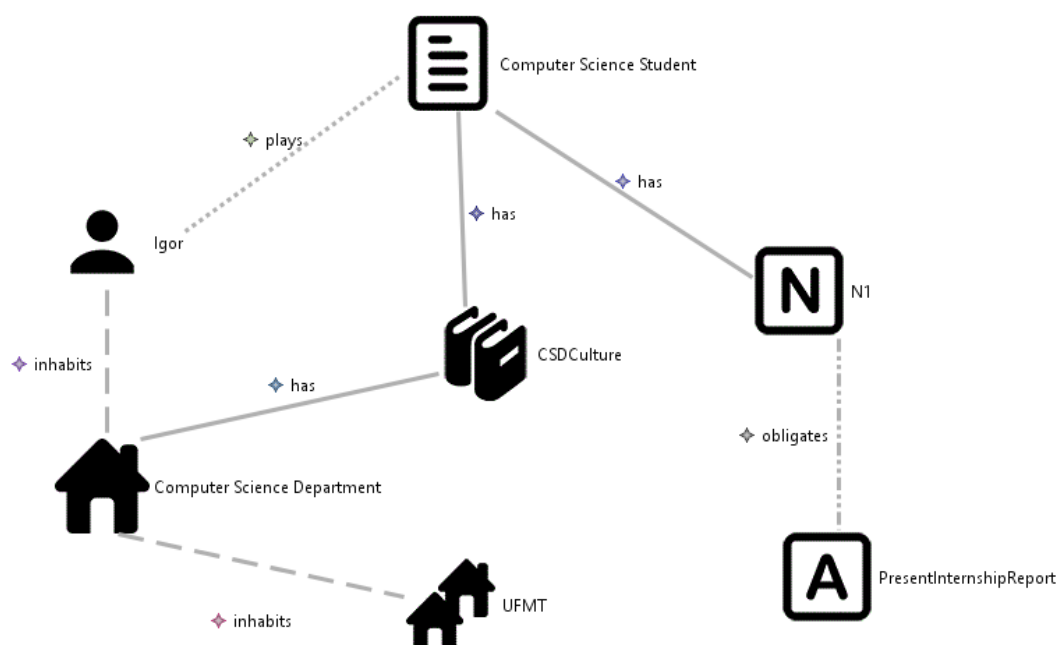


Figura 30 - Exemplo de uma modelagem de um SMA Cultural

4. DIFICULDADES ENCONTRADAS

Ao realizar este trabalho uma das grandes dificuldades iniciais foi a falta de conhecimentos na área de agentes e SMA, tendo em vista que o curso de graduação não oferece para os alunos disciplinas abordando tais assuntos. Para superar esta dificuldade foi necessário fazer o estudo individual deste assunto em livros e artigos orientados pela supervisora de estágio prof. MSc. Karen Figueiredo, que também procurou passar parte de seu conhecimento e esclarecer dúvidas relacionadas ao assunto estudado.

Outro problema encontrado foi a falta de conteúdo relacionado a um erro que ocorreu no desenvolvimento da ferramenta: a GMF estava gerando a paleta com os elementos trocados, e. g. quando o usuário selecionava para criar um agente em sua modelagem era criado qualquer outro elemento aleatório. A origem do problema é desconhecida, um dos motivos que pode ter prejudicado achar a solução foi a falta de proficiência do aluno na língua inglesa, o que acaba se tornando mais um problema. Para que a criação de elementos fosse realizada de forma correta foi necessário fazer a alteração manual nos códigos relacionados à paleta, este processo tornou a criação da ferramenta mais trabalhosa, assim solucionando o problema e atingindo o mesmo objetivo.

5. CONCLUSÕES

O trabalho exposto baseia-se em conceitos relacionados a agentes, SMA, ontologias e modelagem além de trabalhar com as ferramentas: IDE Eclipse, Linguagem de programação Java, *Java Development Kit* (JDK) e *Graphical Modeling Framework* (GMF).

Ao realizar este trabalho um de seus objetivos era desenvolver uma ferramenta de modelagem que permita a criação de modelos de SMA Culturais por analistas de sistemas, pois até então não existiam ferramentas com esta finalidade. Pretendia-se também que, através da elaboração deste trabalho o aluno em estágio aprendesse os conceitos em que o mesmo foi fundamentado e as ferramentas utilizadas.

Com o estudo dos conceitos e a utilização das ferramentas citadas a cima foi possível atingir todos os objetivos deste trabalho, o desenvolvimento da ferramenta de modelagem de SMA Culturais e o aprendizado e experiência do aluno em estágio.

Ferramentas de modelagem tem como finalidade manter a organização e o cumprimento dos requisitos na etapa de desenvolvimento do software e os seus modelos criados auxiliam a ter uma visão mais abrangente do funcionamento do sistema. Espera-se que esta nova ferramenta auxilie os analistas de sistemas e estudantes da área de computação em suas criações de modelos de SMA Culturais e que possa ajudar no crescimento deste novo tema, auxiliando na criação de trabalhos futuros relacionados e também realizando melhorias na própria ferramenta tais como: tornar-se um plug-in da IDE Eclipse para facilitar sua utilização e também fazer testes de validação mais rigorosos para garantir a sua eficiência.

6. REFERÊNCIAS BIBLIOGRÁFICAS

BORDINI, Rafael H.; HÜBNER, Jomi F.; WOOLDRIDGE, Michael. **Programming multi-agent systems in AgentSpeak using Jason**. Inglaterra: John Wiley and Sons. 2007.

CAELUM. **Java e Orientação a Objetos**. [s.l.: s.n.], Disponível em: <<https://www.caelum.com.br/apostila-java-orientacao-objetos/>>. Acesso em: 21 jul. 2014.

CHANDRASEKARAN, Balakrishnan; JOSEPHSON, John R.; BENJAMINS, V. Richard. What are ontologies, and why do we need them?. **IEEE Intelligent systems**, v. 14, n. 1, p. 20-26, 1999.

ECLIPSE FOUNDATION. **About the Eclipse Foundation**. [s.l.: s.n.], Disponível em <<http://www.eclipse.org/org/>>. Acesso em: 21 jul. 2014.

ECLIPSE FOUNDATION. **Graphical Modeling Project (GMP)**. [s.l.: s.n.], Disponível em <<http://www.eclipse.org/modeling/gmp/>>. Acesso em: 21 jul. 2014.

GRUBER, Thomas R. A translation approach to portable ontology specifications. **Knowledge acquisition**, v. 5, n. 2, p. 199-220, 1993.

JENNINGS, Nicholas R. Agent-oriented software engineering. In: **Multiple Approaches to Intelligent Systems**. Springer Berlin Heidelberg, 1999. p. 4-10.

MARQUES, V. F.; FIGUEIREDO, K. S. Uma Ontologia para a Representação de Sistemas Multiagentes Culturais. In WORKSHOP-ESCOLA DE SISTEMAS DE AGENTES, SEUS AMBIENTES E APLICAÇÕES, 8. 2014. Porto Alegre. **Anais...** Porto Alegre: PUCRS, 2014. p. 149-154.

OMG AGENT PLATFORM SPECIAL INTEREST GROUP. **Agent Technology Glossary**. [s.l.: s.n.], Disponível em <<http://www.objs.com/agent/agent-glossary-v02.html>>. Acesso em: 25 jul. 2014.

ORACLE. JAVA™ PLATFORM, **Standard Edition 8 Development Kit**. [s.l.: s.n.], Disponível em <<http://www.oracle.com/technetwork/java/javase/jdk-8-readme-2095712.html>>. Acesso em: 21 jul. 2014.

PRESSMAN, Roger S. **Software engineering: a practitioner's approach**. Nova Iorque: McGraw-Hill, Inc. 1992.

SOMMERVILLE, Ian. **Engenharia de software**. [s.l.]: Addison Wesley, 2003.

VAN GIGCH, John P. **System design modeling and metamodeling**. Nova Iorque: Plenum Press, 1991.