5
Exemplos de aplicações

Com o objetivo de exemplificar a utilização do processo proposto na Seção anterior para o desenvolvimento de SMA, são apresentados dois exemplos de aplicações deste tipo de sistema. O primeiro exemplo (*Virtual MarketPlace*) se refere a uma aplicação de comércio eletrônico. Tais aplicações são consideradas *brenchmarks* no domínio de sistemas multi-agentes [56, 57, 58]. Neste exemplo apresentaremos um exemplo simples de uma loja virtual. O outro exemplo de aplicação (*Expert Committee*) é um SMA para suporte ao gerenciamento de submissões e revisões de artigos submetidos a uma conferência ou workshop, onde os agentes são usados para assistir os pesquisadores e a comissão de organização de um evento em tarefas que fazem parte do processo de revisão e submissão que podem ser automatizadas.

5.1.
Virtual MarketPlace

O primeiro estudo de caso escolhido para exemplificar o processo de desenvolvimento proposto é denominado *Virtual MarketPlace*. Esse sistema implementa uma loja virtual para venda de itens pela internet. Nesta loja, usuários cadastrados podem comprar itens existentes na loja, caso o preço do item na loja seja menor ou igual ao preço que ele deseja pagar.

Ao entrar na loja, o usuário adquire o papel de Comprador, procura por um vendedor e informa a ele o item que deseja comprar. O vendedor, criado pela loja virtual para negociar com o possível comprador, é responsável por verificar se o item existe no ambiente (mercado virtual) e, caso exista, oferecê-lo ao comprador, informando o preço do mesmo. O ambiente armazena todos os itens disponíveis na loja. O usuário poderá aceitar ou não o preço oferecido pelo vendedor. Caso ele aceite, o usuário deverá informar isto ao vendedor, pagá-lo e o vendedor por sua vez deverá informar a loja sobre a venda efetuada. Caso o preço oferecido pelo vendedor seja acima do esperado pelo usuário a compra não é efetuada.

No processo de desenvolvimento temos como primeira etapa a modelagem da aplicação utilizando a linguagem MAS-ML. A Figura 9 abaixo ilustra o diagrama de organização da aplicação em questão. Nele são descritos a organização que representa a Loja Virtual, os agentes do sistema, os papéis que eles podem desempenhar, os objetos e os papéis dos objetos. Na loja em questão podem existir dois tipos de agentes, os agentes que representam os usuários e os que representam os lojistas. Os usuários desempenham o papel de comprador quando desejam adquirir um item da loja. Os lojistas desempenham o papel de vendedor no momento em que interagem com o comprador para vender um item. Através do diagrama é possível perceber que o Mercado Virtual é visto como um ambiente passivo no qual a loja está inserida e onde compradores e vendedores irão negociar os itens. Neste ambiente são armazenados a lista de usuários cadastrados, de seus vendedores e dos itens que são negociados.

Os compradores e os vendedores possuem diferentes interpretações para os itens que estão negociando. Um item é objeto de desejo do comprador e é uma oferta para o vendedor. Estas interpretações diferentes caracterizam os diferentes papéis dos objetos associados ao item. É importante salientar que o diagrama da Figura 9 está representado de forma simplificada, visto que estão suprimidos os compartimentos central e inferior de cada elemento onde são detalhadas as propriedades estáticas e dinâmicas, respectivamente.

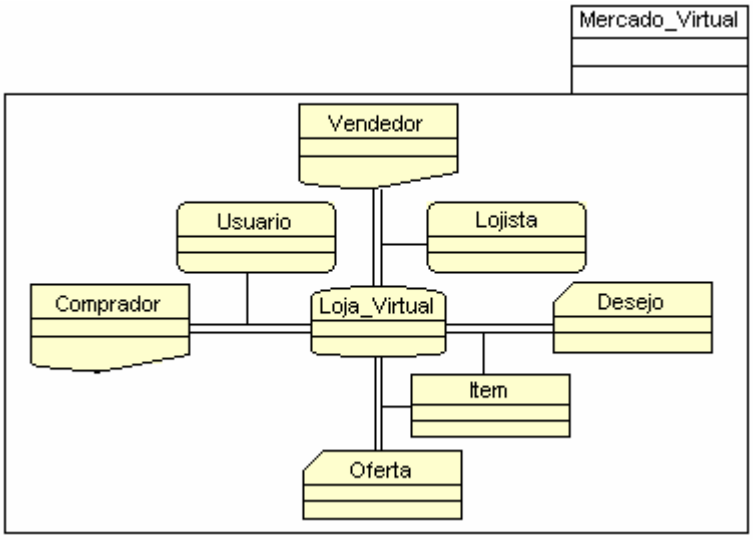


Figura 9 - Diagrama de Organização da aplicação Virtual MarketPlace

Na Figura 10 representamos de forma mais detalhada a Loja Virtual. As principais metas da organização são o gerenciamento dos vendedores e dos itens disponíveis para serem vendidos. Para poder gerenciar os vendedores, esta organização necessita conhecer todos os vendedores que estão

negociando, seus compradores e os itens envolvidos. Este conhecimento é representado por crenças conforme ilustrado na Figura 10. É definido também um axioma para garantir que esta organização informará ao ambiente sobre as vendas. Para alcançar seus objetivos são definidos os seguintes planos para a organização: criação de vendedores para negociar com compradores (Criando_Vendedores) e atualização do ambiente com a situação dos itens (Informando_Venda). O plano responsável por criar os vendedores tem como ação a própria criação do vendedor e informar ao comprador sobre a criação do mesmo.

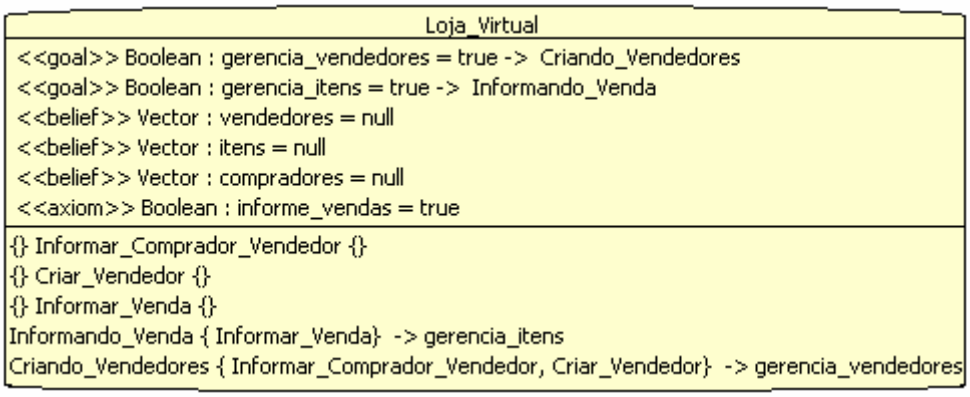


Figura 10 – Representação gráfica em MAS-ML da Loja Virtual

A organização define os papéis de Comprador e Vendedor. O Comprador tem como objetivos procurar o item na loja e comprá-lo, caso o preço o satisfaça. O Vendedor tem o objetivo de vender itens. Tais papéis definem diferentes deveres, direitos e protocolos. O Comprador define com o Vendedor um protocolo simples de negociação. O Comprador tem o direito de aceitar ou rejeitar a oferta feita pelo Vendedor e como dever procurar o Vendedor para a negociação. O papel de Vendedor tem como deveres oferecer o item desejado ao Comprador, informando seu preço e caso a compra seja efetuada, informar a loja. A Figura 11 apresenta a representação gráfica dos papéis de Comprador e Vendedor.

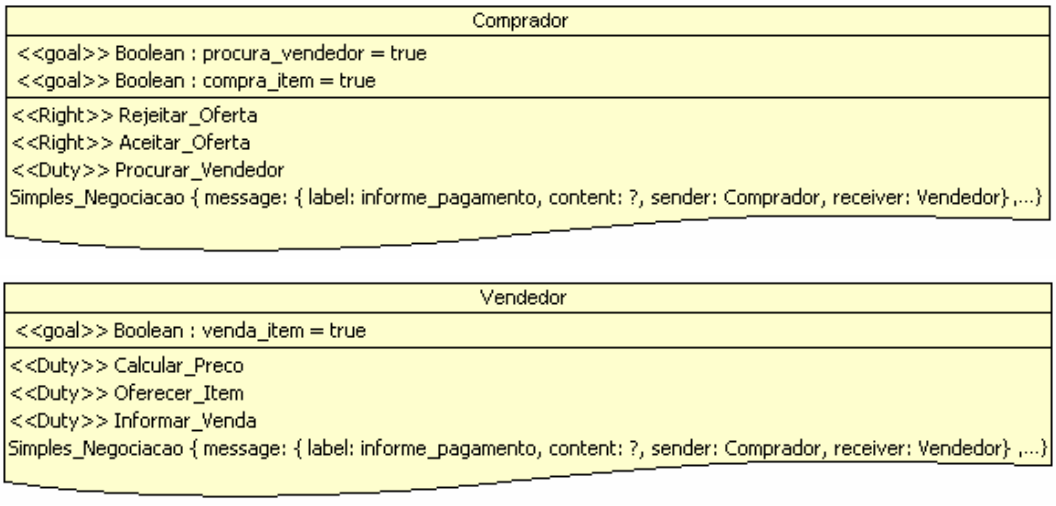


Figura 11 – Representação gráfica em MAS-ML do Comprador e Vendedor

É importante detalhar também os agentes envolvidos nesta aplicação. De acordo com o que foi mencionado, esta aplicação possui dois tipos de agentes, usuário e lojista. O agente Usuário (Figura 12) é criado quando um novo usuário tem por objetivo comprar o item desejado na loja . Para alcançar seu objetivo, este agente deve desempenhar o papel de Comprador mencionado no parágrafo anterior. Os objetivos do Usuário são procurar um vendedor na loja com quem irá negociar e comprar o item desejado. Para alcançar seus objetivos, ele deverá exercer os seguintes planos: procurar um vendedor (Procurando_Vendedor) e negociar o item que deseja comprar (Comprando_Item). Para o plano responsável por negociar o item ele poderá exercer as seguintes ações: procurar o item na loja, avaliar a proposta feita pelo Vendedor, aceitar oferta feita e pagar o item (caso o preço do item esteja dentro do esperado) ou rejeitar a oferta.

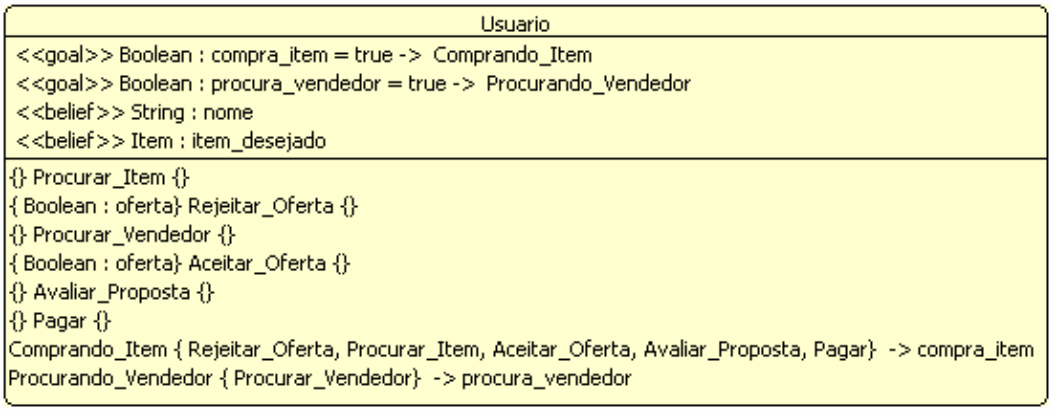


Figura 12 – Representação gráfica em MAS-ML do Usuário

O agente Lojista representa as preferências da loja. Tem como único objetivo vender itens. Este agente deve desempenhar o papel de Vendedor para alcançar seu objetivo. Para isto, deverá executar os seguintes planos: vender o item e informar a organização sobre a venda (Figura 13).

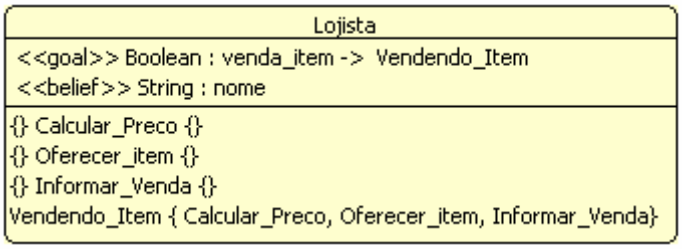


Figura 13 – Representação gráfica em MAS-ML do Lojista

A Figura 14 apresenta os atributos e métodos referentes ao item vendido na loja. Pelo fato dos compradores e vendedores terem diferentes visões sobre o item que desejam negociar, um item pode desempenhar diferentes papéis. Um item é objeto de desejo do comprador e é uma oferta para o vendedor, caracterizando assim diferentes papéis de objetos associados ao item. Apesar de na figura ambos apresentarem os mesmos atributos, na visão de cada papel eles tem valores diferentes, uma vez que os métodos são diferentes. Através do exemplo apresentado na figura abaixo, o papel de objeto denominado Oferta é capaz de atualizar o preço de um item através do método setPreco, porém no papel de objeto Desejo este método não existe uma vez que o Comprador não tem esta permissão.

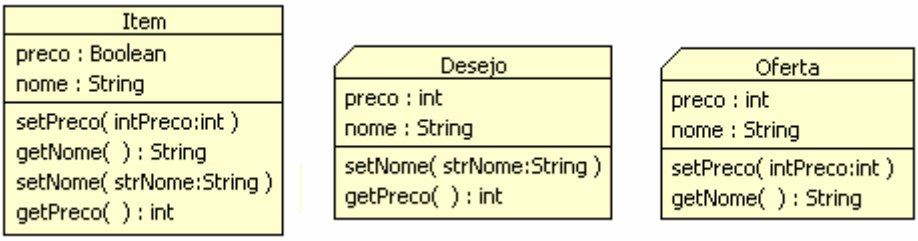


Figura 14 – Representação gráfica em MAS-ML do Item, Desejo e Oferta

Como início do processo de desenvolvimento, a modelagem de MAS-ML, apresentada acima, é transformada no formato MAS-ML XMI. A seguir está representado o trecho correspondente ao diagrama da Figura 9. Nela é possível visualizar os elementos e relacionamentos do diagrama.

```
<UML:Model isAbstract="false" name="Virtual_MarketPlace" xmi.id="S.1">
   <UML:Namespace.ownedElement>
      <UML:Class name="Item" namespace="S.1" xmi.id="S.2">...
      <MASML:AgentClass name="Lojista" namespace="S.1" xmi.id="S.3">...
      <MASML:AgentClass name="Usuario" namespace="S.1" xmi.id="S.4">...
      <MASML:OrganizationClass isMain="true" name="Loja_Virtual" namespace="S.1"
xmi.id="S.5">...
      <MASML:ObjectRoleClass name="Oferta" namespace="S.1" xmi.id="S.6">...
      <MASML:ObjectRoleClass name="Desejo" namespace="S.1" xmi.id="S.7">...
      <MASML:AgentRoleClass name="Comprador" namespace="S.1" xmi.id="S.8">...
      <MASML:AgentRoleClass name="Vendedor" namespace="S.1" xmi.id="S.9">...
      <MASML:PassiveEnvironmentClass name="Mercado_Virtual" namespace="S.1"
xmi.id="S.10">...
      <MASML:Play member="S.2" name="" played="S.7" player="S.5" xmi.id="L.1"/>
      <MASML:Play member="S.2" name="" played="S.6" player="S.5" xmi.id="L.2"/>
      <MASML:Play member="S.4" name="" played="S.8" player="S.5" xmi.id="L.3"/>
      <MASML:Play member="S.3" name="" played="S.9" player="S.5" xmi.id="L.4"/>
      <MASML:Inhabit citzen="S.5" environment="S.10" name="" xmi.id="L.5"/>
```

```
      <MASML:Inhabit citzen="S.4" environment="S.10" name="" xmi.id="L.6"/>
      <MASML:Inhabit citzen="S.3" environment="S.10" name="" xmi.id="L.7"/>
      <MASML:Inhabit citzen="S.2" environment="S.10" name="" xmi.id="L.8"/>
   </UML:Namespace.ownedElement>
</UML:Model>
<MASML:Diagram name="Diagrama_de_Organizacao" xmi.id="D.1">
   <MASML:GraphElement.semanticModel>
      <MASML:SimpleSemanticModelElement typeInfo="OrganizationDiagram"/>
   </MASML:GraphElement.semanticModel>
      ...
      <MASML:PassiveEnvironmentClass xmi.idref="S.10"/>...
      <MASML:OrganizationClass xmi.idref="S.5"/>...
      <MASML:AgentClass xmi.idref="S.3"/>...
      <UML:Class xmi.idref="S.2"/>...
      <MASML:ObjectRoleClass xmi.idref="S.7"/>...
      <MASML:ObjectRoleClass xmi.idref="S.6"/>...
      <MASML:AgentRoleClass xmi.idref="S.8"/>...
      <MASML:AgentRoleClass xmi.idref="S.9"/>...
      <MASML:Inhabit xmi.idref="L.5"/>...
      <MASML:Inhabit xmi.idref="L.6"/>...
      <MASML:Inhabit xmi.idref="L.7"/>...
      <MASML:Inhabit xmi.idref="L.8"/>...
      <MASML:Play xmi.idref="L.1"/>...
      <MASML:Play xmi.idref="L.2"/>...
      <MASML:Play xmi.idref="L.3"/>...
      <MASML:Play xmi.idref="L.4"/>...
      <MASML:AgentClass xmi.idref="S.4"/>...
      ...
</MASML:Diagram>
```

Quadro 12 – Representação em MAS-ML XMI do Diagrama de Organização

O trecho de MAS-ML XMI apresentado a seguir (Quadro 13) contém a representação de alguns dos elementos ilustrados no quadro anterior, com algumas propriedades estão omitidas em virtude da extensão deste documento. O MAS-ML XMI completo está apresentado no Anexo II.

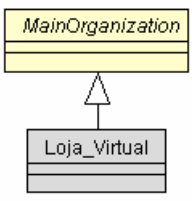
```
<MASML:OrganizationClass isMain="true" name="Loja_Virtual" xmi.id="S.5">
 <UML:Classifier.feature>
  <UML:Attribute name="gerencia_itens" stereotype="SS.1" xmi.id="S.36">
   <UML:Attribute.initialValue>
    <UML:Expression body="true"/>
   </UML:Attribute.initialValue>
   <UML:StructuralFeature.type>
    <UML:Classifier>
     <UML:Namespace.ownedElement>
      <UML:DataType xmi.idref="G.1"/>
     </UML:Namespace.ownedElement>
    </UML:Classifier>
   </UML:StructuralFeature.type>
  </UML:Attribute>
  <UML:Attribute name="vendedores" stereotype="SS.3" xmi.id="S.37">...
  <UML:Attribute name="informe_vendas" stereotype="SS.5" xmi.id="S.40">...
  <MASML:AgentAction name="Informar Venda" xmi.id="S.43"/>...
  <MASML:AgentPlan name="Informando_Venda" xmi.id="S.44">
   <MASML:AgentPlan.actions>
    <Action xmi.idref="S.43"/>
   </MASML:AgentPlan.actions>
   <MASML:AgentPlan.goal stereotype="SS.1" xmi.idref="S.36"/>
  </MASML:AgentPlan>
 </UML:Classifier.feature>
</MASML:OrganizationClass>
<MASML:AgentRoleClass name="Comprador" xmi.id="S.8">
 <UML:Classifier.feature>...
  <MASML:AgentAction name="Aceitar_Oferta" stereotype="SS.7" xmi.id="S.57"/>...
  <MASML:AgentAction name="Procurar Vendedor" stereotype="SS.9" xmi.id="S.58"/>
  <MASML:AgentProtocol name="Simples_Negociacao" xmi.id="S.59">
   <MASML:AgentProtocol.messages>
    <MASML:AgentMessage receiver="S.9" sender="S.8">
     <MASML:AgentMessage.label>informe_pagamento
     </MASML:AgentMessage.label>
     <MASML:AgentMessage.content>?</MASML:AgentMessage.content>
    </MASML:AgentMessage>...
   </MASML:AgentProtocol.messages>
```

```
  </MASML:AgentProtocol>
 </UML:Classifier.feature>
</MASML:AgentRoleClass>
<MASML:ObjectRoleClass isAbstract="false" name="Desejo" xmi.id="S.7">
 <UML:Classifier.feature>...
  <UML:Operation name="getPreco" visibility="public" xmi.id="S.53">
   <UML:BehavioralFeature.parameter>
    <UML:Parameter kind="return" name="getPreco.ret" type="G.3" xmi.id="XX.12"/>
   </UML:BehavioralFeature.parameter>
  </UML:Operation>…
 </UML:Classifier.feature>
</MASML:ObjectRoleClass>
<UML:Stereotype name="Axiom" xmi.id="SS.5">
 <UML:Stereotype.baseClass>S.40</UML:Stereotype.baseClass>
</UML:Stereotype>
<UML:Stereotype name="Goal" xmi.id="SS.1">
 <UML:Stereotype.baseClass>S.36</UML:Stereotype.baseClass>
</UML:Stereotype>
<UML:Stereotype name="Belief" xmi.id="SS.3">
 <UML:Stereotype.baseClass>S.37</UML:Stereotype.baseClass>
</UML:Stereotype>
<UML:Stereotype name="Duty" xmi.id="SS.9">
 <UML:Stereotype.baseClass>S.58</UML:Stereotype.baseClass>
</UML:Stereotype>
<UML:Stereotype name="Right" xmi.id="SS.7">
 <UML:Stereotype.baseClass>S.57</UML:Stereotype.baseClass>
</UML:Stereotype>
```

Quadro 13 – Alguns elementos e propriedades em MAS-ML XMI do Virtual MarketPlace

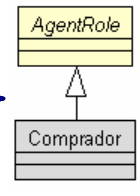
Após a geração do MAS-ML XMI, o mesmo é transformado em UML XMI, com base no framework ASF, conforme apresentado na Seção 4.3.2. O elementos Loja_Virtual e Comprador em UML XMI estão apresentados abaixo. No Anexo II está o documento UML XMI do exemplo.

```
<UML:Class isAbstract="true" name="MainOrganization" xmi.id="F.670">…</UML:Class>
<UML:Class isAbstract="false" name="Loja_Virtual" xmi.id="A.75">
 <UML:Classifier.feature>
  <UML:Operation isAbstract="false" name="Loja_Virtual" xmi.id="A.76">
   <UML:BehavioralFeature.parameter>
    <UML:Parameter kind="return" name="return" type="H.3" xmi.id="A.77"/>
    <UML:Parameter kind="in" name="theEnvironment" type="H.16" xmi.id="A.78"/>
   </UML:BehavioralFeature.parameter>
  </UML:Operation>
  <UML:Method xmi.id="A.79">
   <UML:Method.body>
    <UML:ProcedureExpression body="
     LeafGoal objectGoal = new LeafGoal (“Boolean”,”gerencia_vendedores”,”true”);
     objectGoal.setPlan(“Criando Vendedores”);
     LeafBelief objectBelief = new LeafBelief (“Vector”,”vendedores”,”null”);
     Axiom objectAxiom = new Axiom (“Boolean”,”informe_vendas”, “true”);
     Action action = new Informar Venda();
     Plan objectPlan = new Informando_Venda();..." language="java" xmi.id="A.80"/>
   </UML:Method.body>
   <UML:Method.specification>
    <UML:Operation xmi.idref="A.76"/>
   </UML:Method.specification>
  </UML:Method>...
 </UML:Classifier.feature>...
</UML:Class>
<UML:Generalization xmi.id="A.100">
 <UML:Generalization.child>
  <UML:Class xmi.idref="A.75"/>
 </UML:Generalization.child>
 <UML:Generalization.parent>
  <UML:Class xmi.idref="F.670"/>
 </UML:Generalization.parent>
</UML:Generalization>
```




```
<UML:Class isAbstract="true" name="AgentRole" xmi.id="F.1040">...</UML:Class>
<UML:Class isAbstract="false" name="Comprador" xmi.id="A.137">
 <UML:Classifier.feature>
  <UML:Operation isAbstract="false" name="Comprador" xmi.id="A.138">
   <UML:ModelElement.stereotype>
    <UML:Stereotype xmi.idref="T.1"/>
```

```
    </UML:ModelElement.stereotype>
    <UML:BehavioralFeature.parameter>
     <UML:Parameter kind="return" name="return" type="H.3" xmi.id="A.139"/>
     <UML:Parameter kind="in" name="owner" type="H.15" xmi.id="A.140"/>
    </UML:BehavioralFeature.parameter>
   </UML:Operation>
   <UML:Method xmi.id="A.141">
    <UML:Method.body>
     <UML:ProcedureExpression body="
      LeafGoal objectGoal = new LeafGoal ("Boolean","procura_vendedor","true");
      this.rights.add (new Right ("Aceitar_Oferta"));
      this.duties.add (new Duty ("Procurar Vendedor");
      Protocol objectProtocol =  new application.Simples_Negociacao();
      Message objectMessage = newMessage("informe_pagamento" ,"?" ,"Comprador",
"Vendedor");
      objectProtocol.setMessage(objectMessage);..." language="java"
xmi.id="A.142"/>
    </UML:Method.body>
    <UML:Method.specification>
     <UML:Operation xmi.idref="A.138"/>
    </UML:Method.specification>
   </UML:Method>
  </UML:Classifier.feature>...
</UML:Class>
<UML:Generalization xmi.id="A.143">
 <UML:Generalization.child>
  <UML:Class xmi.idref="A.137"/>
 </UML:Generalization.child>
 <UML:Generalization.parent>
  <UML:Class xmi.idref="F.1040"/>
 </UML:Generalization.parent>
</UML:Generalization>
```



Quadro 14 – Loja Virtual e Comprador do exemplo Virtual MarketPlace em UML XMI

No diagrama de classes ilustrado pela Figura 15, estão representadas algumas das classes instanciadas no framework ASF para a representação da aplicação.

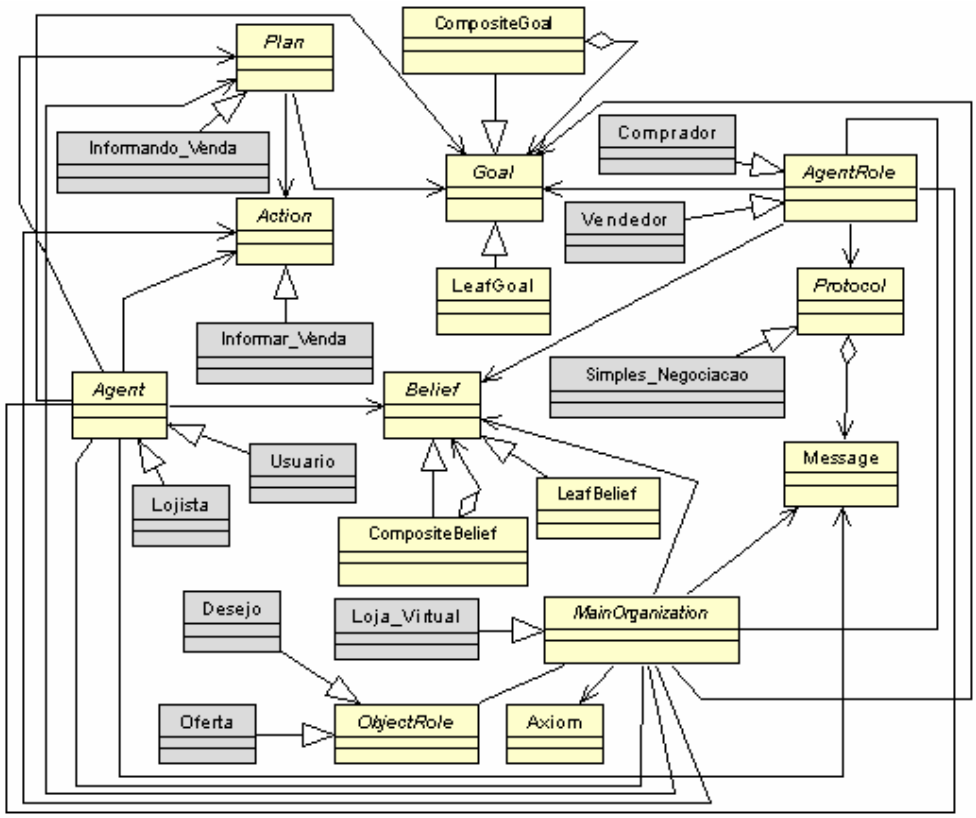


Figura 15 – Diagrama de classes com alguns elementos da Loja Virtual

A última etapa do processo de desenvolvimento proposto é a transformação dos modelos UML, representados em UML XMI, em código. O código apresentado abaixo corresponde aos elementos Loja_Virtual e Comprador que foram representados em UML XMI no Quadro 14. Este código foi gerado utilizando a funcionalidade presente na ferramenta detalhada na Seção 6. Foram omitidas das classes abaixo trechos de código existentes nos seus construtores, como também alguns métodos concretos referentes a métodos abstratos das classes as quais os elementos foram herdados.

```
public class Loja_Virtual extends MainOrganization{
   public Loja_Virtual( Environment theEnvironment, MainOrganization initialOrg,
AgentRole initialRole ){...
      goals = new Vector ();
      LeafGoal objectGoal = new LeafGoal ("Boolean","gerencia_itens","true");
      this.goals.add (objectGoal);
      objectGoal.setPlan("Informando_Venda");
      beliefs = new Vector();
      LeafBelief objectBelief = new LeafBelief ("Vector","vendedores", "null");
      this.beliefs.add (objectBelief);
      objectAxiom = null;
      Axiom objectAxiom = new Axiom ("Boolean","informe_vendas", "true");
      this.axioms.add (objectAxiom);
      actions = new Vector();
      Action action = new Informar_Venda();
      this.actions.add(action);
      Plan objectPlan = new Informando_Venda();
      this.plans.add(objectPlan);
      Action actionAux = null;
      Enumeration enumActions = this.actions.elements();
      while (enumActions.hasMoreElements()){
         actionAux = (Action)enumActions.nextElement();
         if (actionAux.getClass().getName().equals("application.Informar_Venda")){
            objectPlan.setAction(actionAux);
         }
      }...
   }...}
```

Quadro 15 – Código do elemento Loja_Virtual gerado a partir do UML XMI

```
public class Comprador extends AgentRole{
   public Comprador( MainOrganization owner ){...
      goals = new Vector ();
      LeafGoal objectGoal = new LeafGoal ("Boolean","procura_vendedor","true");
      this.goals.add (objectGoal);
      duties = new Vector ();
      rights = new Vector ();
      this.rights.add (new Right ("Aceitar_Oferta"));
      this.duties.add (new Duty ("Procurar_Vendedor"));
      protocols = new Vector ();
      Protocol objectProtocol = new application.Simples_Negociacao();
      this.protocols.add (objectProtocol);
      Message objectMessage = new Message("informe_pagamento","?","Comprador",
"Vendedor");
      objectProtocol.setMessage (objectMessage);...
   }...}
```

Quadro 16 - Código do elemento Comprador gerado a partir do UML XMI

5.2. *Expert Committee*

O Expert Committee é um SMA para suporte ao gerenciamento de submissões e revisões de artigos submetidos a uma conferência ou workshop.

Os agentes criados neste sistema visam assistir os pesquisadores e a comissão de organização de um evento de tarefas que fazem parte deste processo e que podem ser automatizadas. O sistema oferece suporte a diferentes atividades: o envio de trabalhos, a atribuição de um artigo a um revisor, a seleção de revisores, a notificação da aceitação e recusa de artigos, entre outros.

Nesta aplicação, uma Conferência deve ter obrigatoriamente os seguintes personagens: chair, revisores, membros do comitê do programa, coordenador e os autores que irão submeter seus trabalhos na conferência. Todos esses personagens são pesquisadores que podem exercer diferentes papéis neste ambiente (conferência).

Neste tipo de sistema, o Autor envia um Artigo para a Conferência e espera a resposta da revisão. Enquanto isso, o Artigo é recebido pela organização através do Chair. O Chair tem como funções: receber os artigos, distribuí-los entre os revisores, receber seu resultado, enviar o resultado aos autores, analisar os possíveis conflitos, entre outras. O chair deverá organizar os artigos de modo que cada artigo recebido deva ser revisto por no mínimo 3 revisores e cada revisor reveja no máximo 3 artigos. Antes de enviar os artigos aos revisores, o Chair envia uma proposta de revisão, a qual o revisor avalia e informa ao Chair se aceita ou não revisar tais artigos, no caso do revisor aceitar, o Chair envia os artigos a serem revisados. Caso o Chair não tenha revisores suficientes, ele solicita ao Coordenador mais revisores. Caso os resultados de um artigo estejam em conflito, o Chair solicita aos membros análise dos mesmos.

Os membros do comitê devem analisar conflitos que por ventura possam ocorrer em um artigo de acordo com as revisões feitas pelos revisores, retornando ao Chair um voto que indicará se o artigo deve ou não ser aceito. Enquanto o Coordenador visa fornecer revisores ao Chair, caso não existam revisores sufientes e compatíveis com os trabalhos submetidos.

Como parte da primeira etapa do processo de desenvolvimento, temos o diagrama de organização na Figura 16 que contém a organização que representa organização da Conferência, o agente do sitema, os papéis que ele podem desempenhar, o objeto e papel de objeto. Neste ambiente existe apenas um tipo de agente: o Pesquisador. O Pesquisador pode desempenhar diversos papéis nesta conferência e de acordo com cada papel ele terá diferentes responsabilidades.

Caso o Pesquisador participe da conferência através da submissão de seus trabalhos, ele estará desempenhando o papel de Autor. Há também neste

ambiente pesquisadores que tem como função participar da coordenação da Conferência. Dependendo da função que ele exercerá, os possíveis papéis que eles podem desempenhar são: Chair, Membro, Coordenador geral e Revisor. Mais adiante serão apresentadas funções de cada papel de agente.

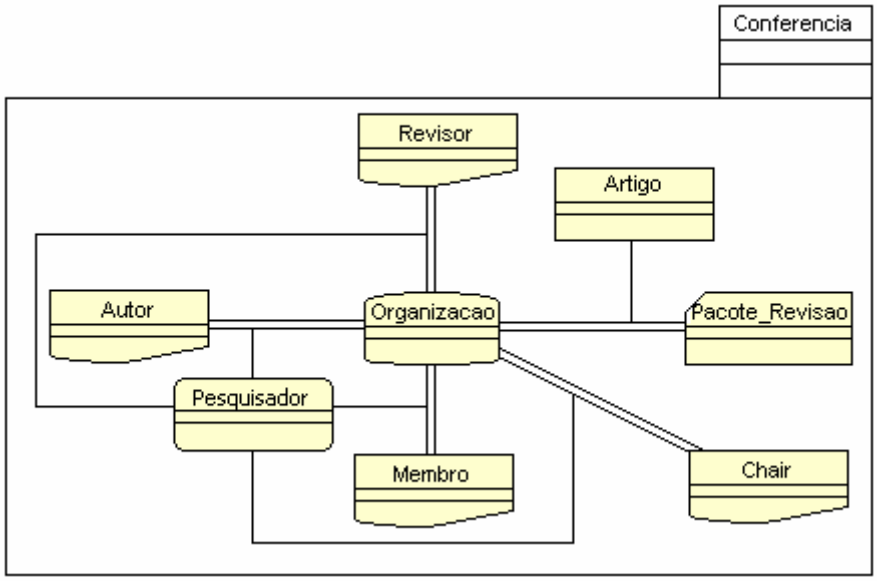


Figura 16 –Diagrama de Organização da aplicação Expert Committee

Na Figura 17 representamos de forma mais detalhada a organização. Sua meta é o gerenciamento dos artigos submetidos. Para isto, esta organização necessita conhecer os artigos enviados pelos autores. Além disso, tem conhecimento das principais datas da Conferência (como por exemplo a data limite para submissão de arquivos, a data limite de resposta da revisão, entre outras), o Chair, seus membros e o Coordenador. Este conhecimento é representado por crenças conforme ilustrado na Figura 17. Para alcançar seus objetivos é definido o plano “Gerenciando_Artigos”.

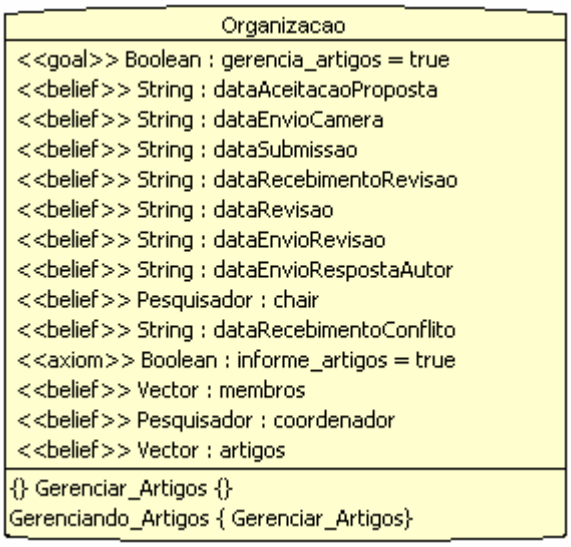


Figura 17 – Representação gráfica em MAS-ML da Organização

A organização define o papel Autor que tem como objetivos submeter o arquivo na Conferência e receber seu resultado, para então submeter o artigo final se for este o caso. O Autor tem o direito de enviar a versão final e como deveres submeter o Artigo e receber seu resutado. A Figura 18 apresenta a representação gráfica do papel de Autor.

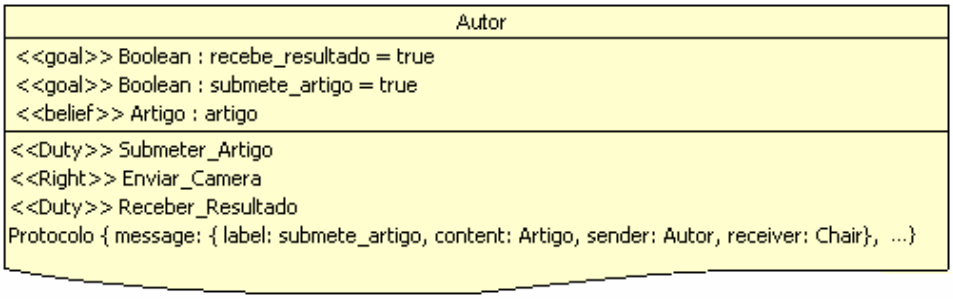


Figura 18 – Representação gráfica em MAS-ML do Autor (papel de agente)

Outro papel importante na organização é o do Chair. Para realizar o seu papel foram definidas metas para atender as funcionalidades exercidas por ele e citadas anteriormente. Como exemplo temos a meta “organiza_artigos”, que tem por objetivo organizar os artigos recebidos e distribuí-los entre os revisores, respeitando as regras já mencionadas. O Chair define com os demais papéis um protocolo de comunicação. O papel de Chair tem como deveres receber os artigos, organizar os mesmos, distribuí-los aos revisores, receber sua revisão, analisar a ocorrência de conflitos e enviar o resultado da revisão ao Autor do artigo. Caso haja conflitos, ele tem o direito de enviar os conflitos aos membros e analisar os votos dos mesmos.

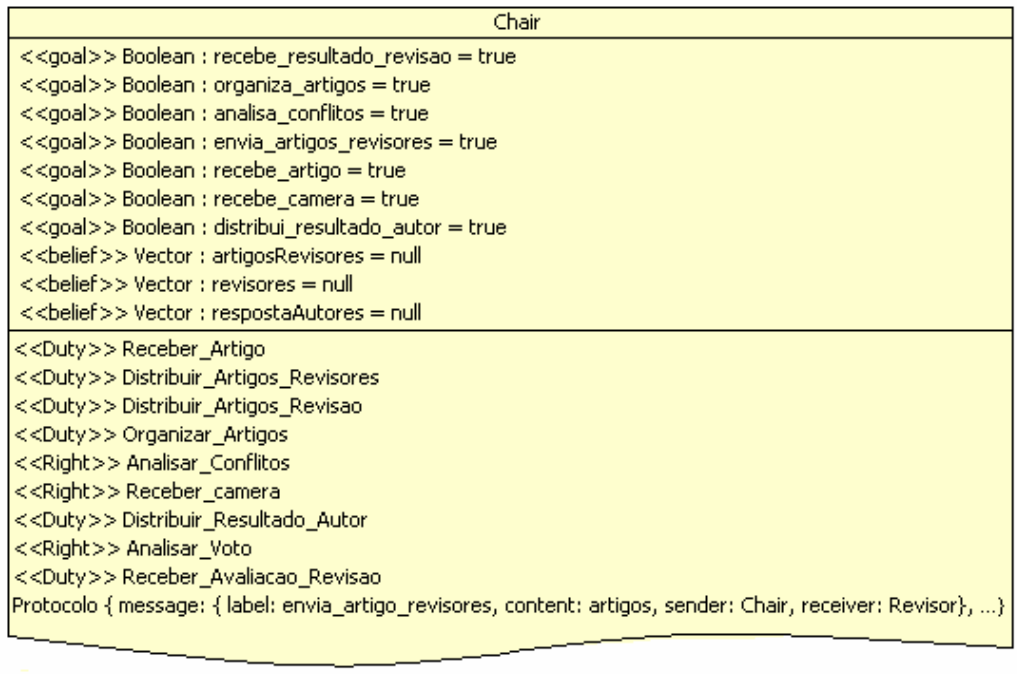


Figura 19 – Representação gráfica em MAS-ML do Chair

O pesquisador que for um membro do comitê do programa terá como meta avaliar os possíveis conflitos que possam ocorrer e votar pela aceitação ou rejeição dos artigos que se encontram neste caso. Este papel está representado pela Figura 20.

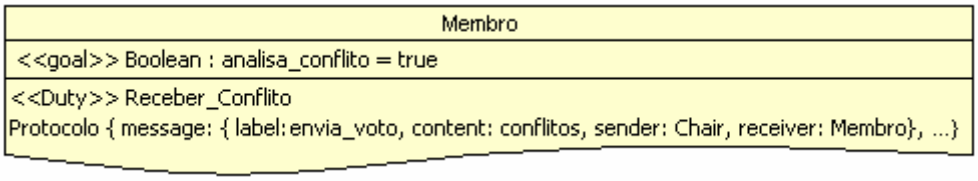


Figura 20 – Representação gráfica em MAS-ML do Membro

Na Figura 21 temos o papel de Revisor que tem como metas analisar a proposta de revisão (proposta_revisão) e receber os artigos a serem revisados (revisão), caso aceite a proposta. A proposta de revisão consiste em um conjunto de no máximo três “pacotes de revisão”.

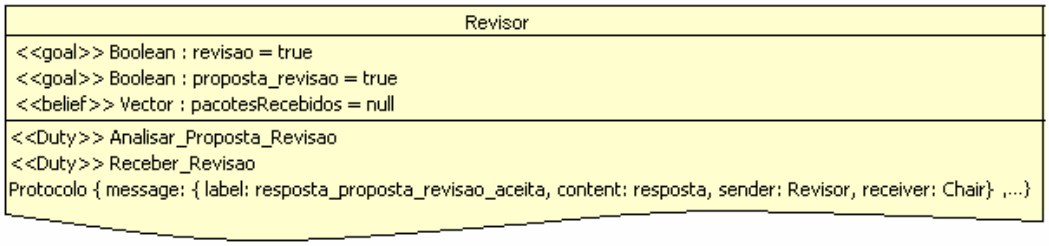


Figura 21 – Representação gráfica em MAS-ML do Revisor

O denominado “Pacote de Revisão” corresponde a uma primeira visão do Revisor sobre um determinado Artigo. Neste pacote estão representadas apenas algumas informações relacionadas ao Artigo, como: título, resumo e áreas envolvidas. Por ser uma visão diferente de um Artigo, o “Pacote de Revisão” foi modelado como um papel de objeto. Na Figura 22 estão detalhados o objeto Artigo e o papel de objeto “Pacote de Revisão”.

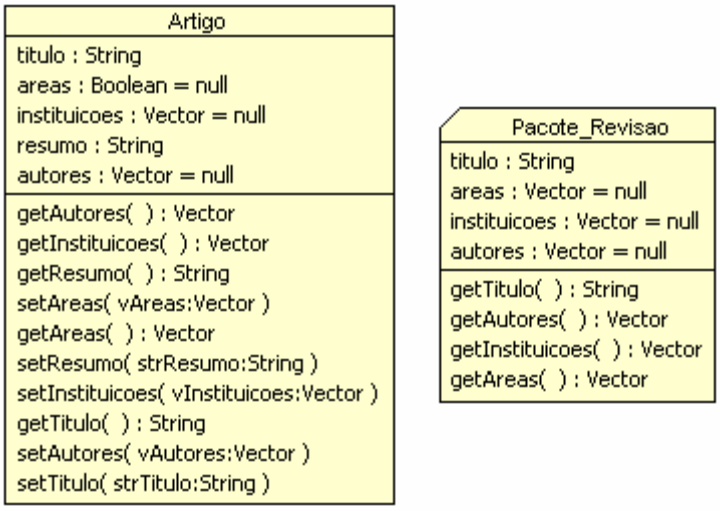


Figura 22 – Representação gráfica em MAS-ML do Artigo e Pacote Revisão

O Coordenador (Figura 23) tem como objetivo fornecer revisores ao Chair de uma conferência. Quando o chair pede novos revisores, informa o número de revisores que deseja e a área de interesse. O Coordenador consulta a sua base e fornece uma lista de novos revisores.

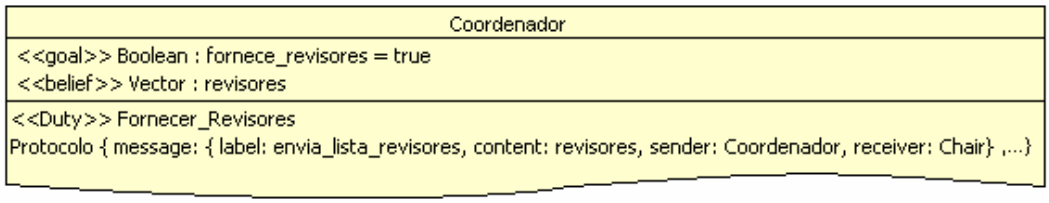


Figura 23 – Representação gráfica em MAS-ML do Coordenador

O agente Pesquisador por sua vez poderá desempenhar qualquer um dos papéis descritos acima. Por este motivo, seu objetivo será de acordo com o papel que desempenhará. Deste modo, na modelagem foi criado um objetivo relacionado a cada papel. Os objetivos a priori estarão desabilitados e de acordo com a função desempenhada pelo agente, caberá a ele habilitar o plano relacionado ao papel que deseja desempenhar.

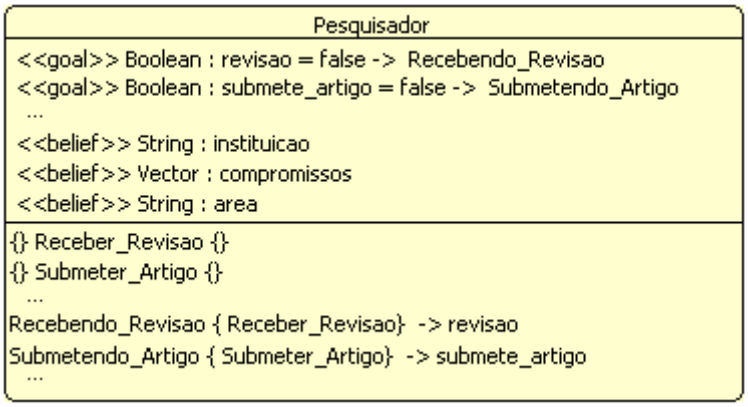


Figura 24 – Representação gráfica em MAS-ML do Pesquisador

A partir da modelagem gráfica apresentada acima é gerado o MAS-ML XMI. No quadro seguinte está apresentado um trecho do MAS-ML XMI correspondente ao diagrama de organização da Figura 16. Nela é possível visualiar os elementos e relacionamentos do diagrama.

```
<UML:Model name="Expert_Committee" xmi.id="S.1">
<UML:Namespace.ownedElement>
 <UML:Class name="Artigo" namespace="S.1" xmi.id="S.2">...
 <MASML:AgentClass name="Pesquisador" namespace="S.1" xmi.id="S.3">...
 <MASML:OrganizationClass name="Organizacao" namespace="S.1" xmi.id="S.4">...
 <MASML:ObjectRoleClass name="Pacote_Revisao" namespace="S.1" xmi.id="S.5">...
 <MASML:AgentRoleClass name="Autor" namespace="S.1" xmi.id="S.6">...
 <MASML:AgentRoleClass name="Coordenador" namespace="S.1" xmi.id="S.7">...
 <MASML:AgentRoleClass name="Membro" namespace="S.1" xmi.id="S.8">...
 <MASML:AgentRoleClass name="Chair" namespace="S.1" xmi.id="S.9">...
 <MASML:AgentRoleClass name="Revisor" namespace="S.1" xmi.id="S.10">...
 <MASML:PassiveEnvironmentClass name="Conferencia" namespace="S.1"
xmi.id="S.11"/>...
 <MASML:Play member="S.3" name="" played="S.8" player="S.4" xmi.id="L.1"/>...
 <MASML:Play member="S.3" name="" played="S.9" player="S.4" xmi.id="L.2"/>...
 <MASML:Play member="S.3" name="" played="S.6" player="S.4" xmi.id="L.3"/>...
```

```
 <MASML:Play member="S.2" name="" played="S.5" player="S.4" xmi.id="L.4"/>...
 <MASML:Play member="S.3" name="" played="S.7" player="S.4" xmi.id="L.5"/>...
 <MASML:Play member="S.3" name="" played="S.10" player="S.4" xmi.id="L.6"/>...
 <MASML:Inhabit citzen="S.4" environment="S.11" name="" xmi.id="L.7"/>...
 <MASML:Inhabit citzen="S.2" environment="S.11" name="" xmi.id="L.8"/>...
 <MASML:Inhabit citzen="S.3" environment="S.11" name="" xmi.id="L.9"/>...
</UML:Namespace.ownedElement>
</UML:Model>
<MASML:Diagram name="Diagrama_Organizacao_1" xmi.id="D.1">
<MASML:GraphElement.semanticModel>
 <MASML:SimpleSemanticModelElement typeInfo="OrganizationDiagram"/>
</MASML:GraphElement.semanticModel>...
 <MASML:AgentClass xmi.idref="S.3"/>...
 <UML:Class xmi.idref="S.2"/>...
 <MASML:AgentRoleClass xmi.idref="S.6"/>...
 <MASML:AgentRoleClass xmi.idref="S.9"/>...
 <MASML:AgentRoleClass xmi.idref="S.8"/>...
 <MASML:AgentRoleClass xmi.idref="S.10"/>...
 <MASML:PassiveEnvironmentClass xmi.idref="S.11"/>...
 <MASML:ObjectRoleClass xmi.idref="S.5"/>...
 <MASML:OrganizationClass xmi.idref="S.4"/>...
 <MASML:AgentRoleClass xmi.idref="S.7"/>...
 <MASML:Inhabit xmi.idref="L.9"/>...
 <MASML:Inhabit xmi.idref="L.8"/>...
 <MASML:Inhabit xmi.idref="L.7"/>...
 <MASML:Play xmi.idref="L.4"/>...
 <MASML:Play xmi.idref="L.3"/>...
 <MASML:Play xmi.idref="L.2"/>...
 <MASML:Play xmi.idref="L.1"/>...
 <MASML:Play xmi.idref="L.6"/>...
</MASML:Diagram>
```

Quadro 17 – Representação em MAS-ML XMI do Diagrama de Organização

O trecho do MAS-ML XMI descrito a seguir contém a representação dos elementos Membro e Pesquisador, mencionados anteriormente, sendo que algumas propriedades estão sendo omitidas com o objetivo de não estender demais este quadro. O MAS-ML XMI completo desta aplicação está presente no Anexo III deste documento.

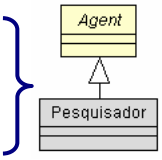
```
<MASML:AgentClass name="Pesquisador" namespace="S.1" xmi.id="S.3">
<UML:Classifier.feature>
 <UML:Attribute name="submete_artigo" stereotype="SS.1" xmi.id="S.36">
  <UML:Attribute.initialValue>
   <UML:Expression body="false"/>
  </UML:Attribute.initialValue>
  <UML:StructuralFeature.type>
   <UML:Classifier>
    <UML:Namespace.ownedElement>
     <UML:DataType xmi.idref="G.1"/>
    </UML:Namespace.ownedElement>
   </UML:Classifier>
  </UML:StructuralFeature.type>
 </UML:Attribute>...
 <UML:Attribute name="area" stereotype="SS.3" xmi.id="S.41">
  <UML:StructuralFeature.type>
   <UML:Classifier>
    <UML:Namespace.ownedElement>
     <UML:DataType xmi.idref="G.2"/>
    </UML:Namespace.ownedElement>
   </UML:Classifier>
  </UML:StructuralFeature.type>
 </UML:Attribute>...
 <MASML:AgentAction name="Submeter_Artigo" xmi.id="S.52"/>...
 <MASML:AgentPlan name="Submetendo_Artigo" xmi.id="S.64">
  <MASML:AgentPlan.actions>
   <Action xmi.idref="S.52"/>
  </MASML:AgentPlan.actions>
  <MASML:AgentPlan.goal stereotype="SS.1" xmi.idref="S.36"/>
 </MASML:AgentPlan>...
</UML:Classifier.feature>
</MASML:AgentClass>
<MASML:AgentRoleClass name="Membro" namespace="S.1" xmi.id="S.8">
```

```
<UML:Classifier.feature>
 <UML:Attribute name="analisa_conflito" stereotype="SS.1" xmi.id="S.106">
  <UML:Attribute.initialValue>
   <UML:Expression body="true"/>
  </UML:Attribute.initialValue>
  <UML:StructuralFeature.type>
   <UML:Classifier>
    <UML:Namespace.ownedElement>
     <UML:DataType xmi.idref="G.1"/>
    </UML:Namespace.ownedElement>
   </UML:Classifier>
  </UML:StructuralFeature.type>
 </UML:Attribute>
 <MASML:AgentAction name="Receber_Conflito" stereotype="SS.7" xmi.id="S.107"/>
 <MASML:AgentProtocol name="Protocolo" xmi.id="S.108">
  <MASML:AgentProtocol.messages>
   <MASML:AgentMessage receiver="S.8" sender="S.9">
    <MASML:AgentMessage.label>envia_conflitos</MASML:AgentMessage.label>
    <MASML:AgentMessage.content>conflitos</MASML:AgentMessage.content>
   </MASML:AgentMessage>...
  </MASML:AgentProtocol.messages>
 </MASML:AgentProtocol>
</UML:Classifier.feature>
</MASML:AgentRoleClass>
<UML:Stereotype name="Duty" xmi.id="SS.7">
 <UML:Stereotype.baseClass>S.107</UML:Stereotype.baseClass>
</UML:Stereotype>
<UML:Stereotype name="Goal" xmi.id="SS.1">
 <UML:Stereotype.baseClass>S.36</UML:Stereotype.baseClass>
 <UML:Stereotype.baseClass>S.106</UML:Stereotype.baseClass>
</UML:Stereotype>
<UML:Stereotype name="Belief" xmi.id="SS.3">
 <UML:Stereotype.baseClass>S.41</UML:Stereotype.baseClass>
</UML:Stereotype>
```

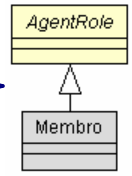
Quadro 18 – Representação em MAS-ML XMI dos elementos Membro e Pesquisador

Após a transformação da representação gráfica em MAS-ML XMI, este é transformado em UML XMI de acordo com o framework ASF. Os elementos Membro e Pesquisador estão apresentados abaixo em UML XMI.

```
<UML:Class isAbstract="true" name="Agent" xmi.id="F.1271">
<UML:Class isAbstract="false" name="Pesquisador" xmi.id="A.46">
<UML:Classifier.feature>
 <UML:Operation isAbstract="false" name="Pesquisador" xmi.id="A.47">
  <UML:BehavioralFeature.parameter>
   <UML:Parameter kind="return" name="return" type="H.3" xmi.id="A.48"/>
   <UML:Parameter kind="in" name="theEnvironment" type="H.16" xmi.id="A.49"/>
   <UML:Parameter kind="in" name="initialOrg" type="H.15" xmi.id="A.50"/>
   <UML:Parameter kind="in" name="initialRole" type="H.14" xmi.id="A.51"/>
  </UML:BehavioralFeature.parameter>
 </UML:Operation>
 <UML:Method xmi.id="A.52">
  <UML:Method.body>
   <UML:ProcedureExpression body="
     LeafGoal objectGoal = new LeafGoal (“Boolean”,”submete_artigo”,”false”);
     objectGoal.setPlan(“Submetendo_Artigo”);
     LeafBelief objectBelief = new LeafBelief (“String”,”area”, “”);
     Action objectAction= new Submeter_Artigo();
     Plan objectPlan = new Submetendo_Artigo();…"
   language="java" xmi.id="A.53"/>
  </UML:Method.body>
  <UML:Method.specification>
   <UML:Operation xmi.idref="A.47"/>
  </UML:Method.specification>
 </UML:Method>...
</UML:Classifier.feature>...
</UML:Class>
<UML:Generalization xmi.id="A.73">
<UML:Generalization.child>
 <UML:Class xmi.idref="A.46"/>
</UML:Generalization.child>
<UML:Generalization.parent>
 <UML:Class xmi.idref="F.1271"/>
</UML:Generalization.parent>
```



```
</UML:Generalization>
<UML:Class isAbstract="true" name="AgentRole" xmi.id="F.1040">
<UML:Class isAbstract="false" name="Membro" xmi.id="A.144">
<UML:Classifier.feature>
 <UML:Operation isAbstract="false" name="Membro" xmi.id="A.145">
  <UML:BehavioralFeature.parameter>
   <UML:Parameter kind="return" name="return" type="H.3" xmi.id="A.146"/>
   <UML:Parameter kind="in" name="owner" type="H.15" xmi.id="A.147"/>
  </UML:BehavioralFeature.parameter>
 </UML:Operation>
 <UML:Method xmi.id="A.148">
  <UML:Method.body>
   <UML:ProcedureExpression body="
    LeafGoal objectGoal = new LeafGoal ("Boolean","analisa_conflito","true");
    this.duties.add (new Duty ("Receber_Conflito"));
    Protocol objectProtocol = new application.Protocolo();
    Message objectMessage = new Message("envia_voto","voto","Membro", "Chair");
    objectProtocol.setMessage (objectMessage); "
    language="java" xmi.id="A.149"/>
  </UML:Method.body>
  <UML:Method.specification>
   <UML:Operation xmi.idref="A.145"/>
  </UML:Method.specification>
 </UML:Method>
</UML:Classifier.feature>
...
</UML:Class>
<UML:Generalization xmi.id="A.150">
<UML:Generalization.child>
 <UML:Class xmi.idref="A.144"/>
</UML:Generalization.child>
<UML:Generalization.parent>
 <UML:Class xmi.idref="F.1040"/>
</UML:Generalization.parent>
</UML:Generalization>
<UML:Class isAbstract="false" name="Submeter_Artigo" xmi.id="A.203">...
<UML:Class isAbstract="false" name="Protocolo" xmi.id="A.281">...
<UML:Class isAbstract="false" name="Submetendo_Artigo" xmi.id="A.301">...
```



Quadro 19 – Membro e Pesquisador do exemplo Expert Committee em UML XMI

O diagrama de classes abaixo (Figura 25) apresenta algumas das classes criadas após a transformação dos modelo MAS-ML em modelo UML, de acordo com o framework ASF.

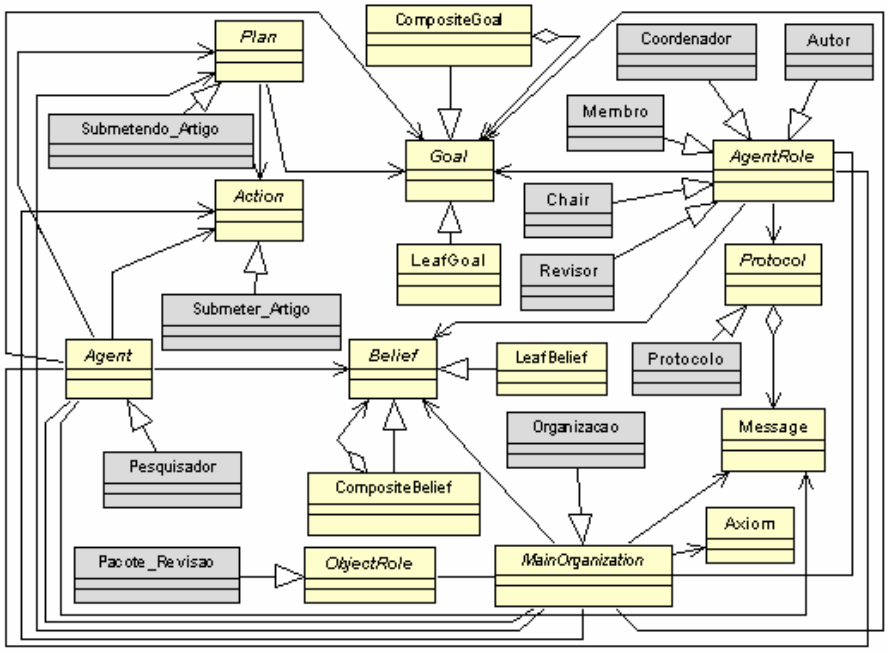


Figura 25 - Diagrama de classes com alguns elementos do Expert Comittee

Finalmente, o UML XMI gerado é transformado em código. O código apresentado abaixo é referente aos elementos Membro e Pesquisador da aplicação. Este código foi gerado utilizando a ferramenta apresentada na Seção 6, sendo desconsiderados alguns trechos dos construtores.

```
public class Membro extends AgentRole{
   public Membro( MainOrganization owner ){
      super.setOwner( owner );
      owner.setAgentRole( this );
      LeafGoal objectGoal = null;
      goals = new Vector ();
      objectGoal = new LeafGoal ("Boolean","analisa_conflito","true");
      this.goals.add (objectGoal);
      LeafBelief objectBelief = null;
      beliefs = new Vector ();
      duties = new Vector ();
      rights = new Vector ();
      this.duties.add (new Duty ("Receber_Conflito"));
      Message objectMessage = null;
      Protocol objectProtocol = null;
      protocols = new Vector ();
      objectProtocol = new application.Protocolo();
      this.protocols.add (objectProtocol);
      objectMessage    =    new    Message("envia_conflitos","conflitos","Chair",
"Membro");
      objectProtocol.setMessage (objectMessage);
      objectMessage = new Message("envia_voto","voto","Membro", "Chair");
      objectProtocol.setMessage (objectMessage);
   }
}
```

Quadro 20 - Código do elemento Membro gerado a partir do UML XMI

```
public class Pesquisador extends Agent{
   public  Pesquisador(  Environment  theEnvironment,  MainOrganization  initialOrg,
AgentRole initialRole ){
      environment = theEnvironment;
      environment.registerAgents( this );
      setRolesBeingPlayed( initialRole, initialOrg );
      LeafGoal objectGoal = null;
      goals = new Vector ();
      objectGoal = new LeafGoal ("Boolean","revisao","false");
      this.goals.add (objectGoal);
      objectGoal.setPlan("Recebendo_Revisao");
      objectGoal = new LeafGoal ("Boolean","submete_artigo","false");
      this.goals.add (objectGoal);
      objectGoal.setPlan("Submetendo_Artigo");...
      LeafBelief objectBelief = null;
      beliefs = new Vector ();
      objectBelief = new LeafBelief ("String","instituicao", "");
      this.beliefs.add (objectBelief);
      objectBelief = new LeafBelief ("Vector","compromissos", "");
      this.beliefs.add (objectBelief);
      objectBelief = new LeafBelief ("String","area", "");
      this.beliefs.add (objectBelief);
      Condition objectCondition = null;
      Action objectAction= null;
      actions = new Vector();
      objectAction = new Receber_Artigo();
      this.actions.add(objectAction);
      objectAction = new Submeter_Artigo();
      this.actions.add(objectAction);...
      Goal goalAux = null;
      Action actionAux = null;
      Enumeration enumActions = null;
      Enumeration enumGoals = null;
      Plan objectPlan = null;
      plans = new Vector();
      objectPlan = new Recebendo_Revisao();
      this.plans.add(objectPlan);
      actionAux = null;
      enumActions = this.actions.elements();
```

```
      while (enumActions.hasMoreElements()){
         actionAux = (Action)enumActions.nextElement();
         if
(actionAux.getClass().getName().equals("application.Receber_Revisao")){
            objectPlan.setAction(actionAux);
         }
      }...
      objectPlan = new Submetendo_Artigo();
      this.plans.add(objectPlan);
      actionAux = null;
      enumActions = this.actions.elements();
      while (enumActions.hasMoreElements()){
         actionAux = (Action)enumActions.nextElement();
         if
(actionAux.getClass().getName().equals("application.Submeter_Artigo")){
            objectPlan.setAction(actionAux);
         }
      }...
   }...
}
```

Quadro 21 - Código do elemento Pesquisador gerado a partir do UML XMI