

4 Processo de Transformação

Com a constante mudança nos requisitos (funcionais e não funcionais) do domínio da aplicação, há uma grande necessidade de que os sistemas estejam aptos a se adaptarem a essas mudanças.

A abordagem MDA é muito utilizada para solucionar problemas desse tipo, possibilitando a automatização de alguns passos que antes deveriam ser feitos manualmente. Por esse motivo o processo de transformação proposto nesta dissertação, utiliza a abordagem MDA.

O processo visa diminuir o trabalho manual, durante a etapa de geração e de adaptação de modelos a diferentes tecnologias. Então caso seja necessário adaptar um modelo independente de plataforma (PIM) para usar um framework de persistência, o processo dará suporte para que um novo modelo PSM seja gerado, a partir das informações necessárias para a adaptação dos mesmos.

Vale ressaltar que o processo proposto nessa dissertação é totalmente voltado para a camada de persistência, diminuindo o trabalho manual durante a inclusão de Frameworks, Patterns e tecnologias relacionadas à camada de persistência. A aplicação do processo proposto às outras camadas deverá ser alvo de estudos posteriores, para que seja comprovada a sua visibilidade.

Como o processo é baseado na abordagem MDA, é necessário ter um modelo independente de plataforma (PIM), e o mesmo deve estar descrito na forma de um ou mais diagramas de classes da UML [UML]. Esse modelo alto nível será trabalhado durante o processo proposto, com o intuito de gerar um novo modelo adaptado às tecnologias de persistência.

A figura 26, a seguir, dá uma visão geral do processo proposto nesta dissertação.

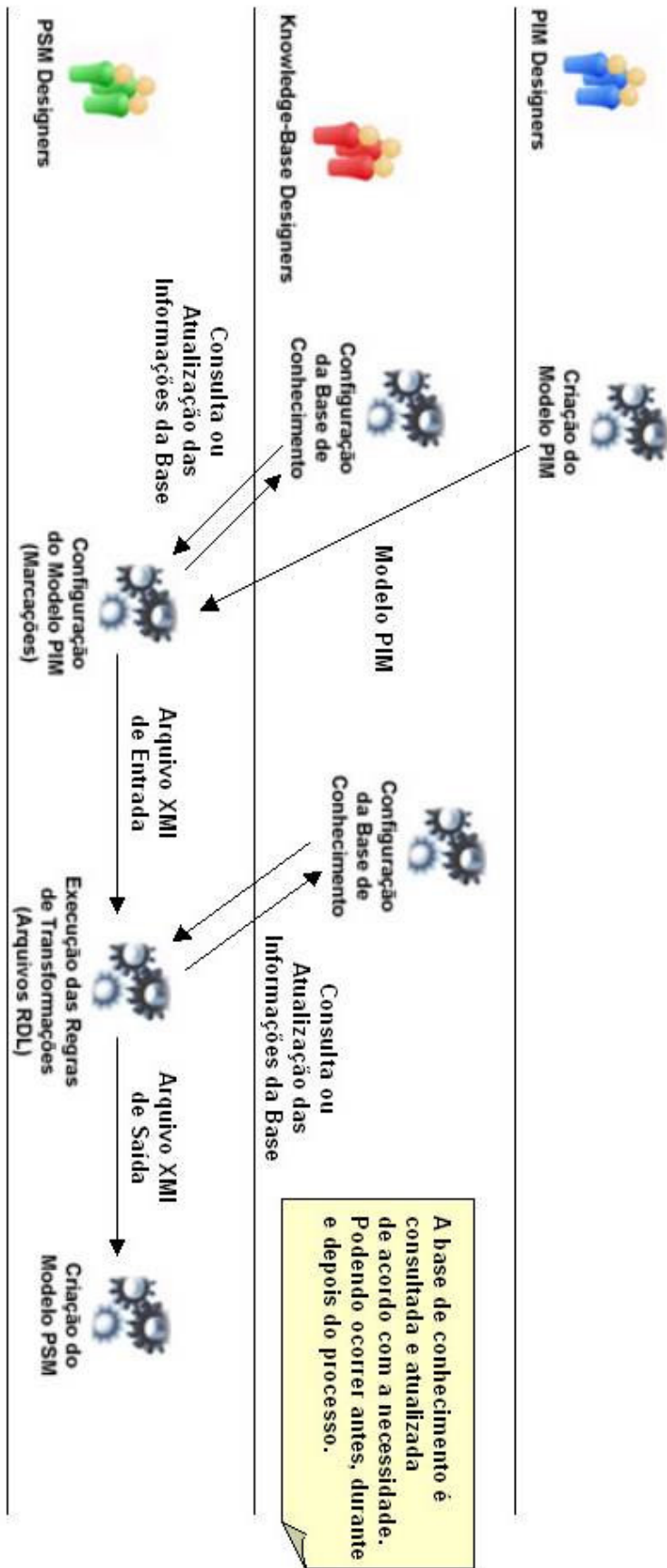


Figura 26 - Visão Geral do Processo Proposto

Todas as etapas vistas na figura 26 serão descritas detalhadamente nas seções seguintes.

Foi introduzido no processo o conceito de base de conhecimento, tendo como objetivo principal mapear as informações do modelo com as regras de transformações a serem realizadas no mesmo. Essas informações são adicionadas na forma de marcações, sendo as mesmas representadas através de estereótipos [UML] nas classes do diagrama UML.

As regras de transformações estão armazenadas na forma de arquivos RDL, que descrevem o passo a passo da mesma de forma organizada e formalizada. É possível criar novas transformações e adicioná-las na base de conhecimento do processo.

O resultado do processo é um modelo específico para uma plataforma, adaptado a tecnologias da camada de persistência, podendo ser Frameworks, Patterns, entre outros. Alguns arquivos de configuração específicos para alguns tipos de Frameworks de Persistência, também poderão ser gerados.

Os PIM Designers são os responsáveis pela criação e manutenção do modelo PIM. É necessário que os mesmos tenham uma visão do domínio da aplicação, pois esse modelo possui um maior nível de abstração, e deve ser independente de plataformas e tecnologias. Já os Knowledge-Base Designers são os responsáveis pela manutenção da base de conhecimento. Os mesmos devem ter conhecimento das possíveis tecnologias utilizadas no desenvolvimento, e incluí-las, caso necessário, na base. Quanto mais informações da base puderem ser utilizadas durante o processo, menor será o trabalho manual necessário. E por último os PSM Designers são os responsáveis por todas as etapas posteriores a criação do PIM, necessárias para a geração do Modelo PSM.

4.1. Criação do Modelo PIM

O processo tem como entrada um modelo independente de plataforma, sendo que o mesmo já deve estar configurado com as marcações. Isso é necessário, pois é a partir delas que o processo poderá definir quais regras de transformações poderão ser utilizadas no mesmo.

O modelo deve ser desenvolvido utilizando o diagrama de classes da UML, pois as transformações e a geração do novo modelo, ocorrem neste mesmo tipo de diagrama. A figura 27 mostra um modelo PIM, que será utilizado para ilustrar as demais etapas do processo.

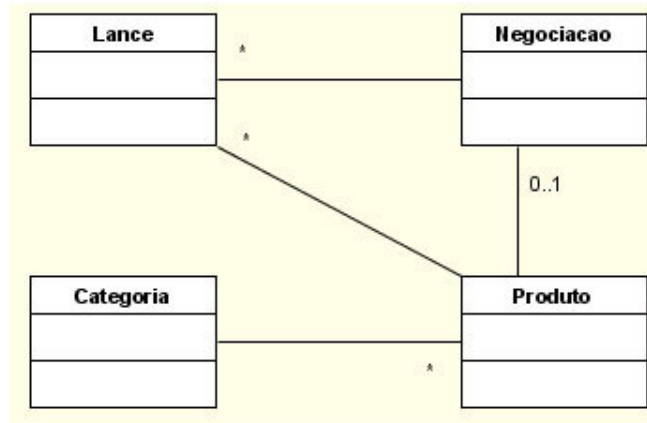


Figura 27 - Modelo PIM

4.2. Configuração do Modelo PIM (Marcações)

As marcações são a forma de indicar quais classes deverão ser adaptadas para a nova plataforma, e também qual o tipo de transformação que deverá ocorrer no modelo. Elas são representadas na forma de esteriótipos UML associados às classes de um modelo PIM.

Elas estão diretamente ligadas a base de conhecimento, obrigando as mesmas a serem anteriormente cadastradas, para que possam ser utilizadas como marcações no modelo. O Pim Designer terá que ter conhecimento das informações contidas na base de conhecimento, para que consiga configurar o modelo corretamente.

No processo proposto as marcações utilizadas nos modelos são as de alto nível (independentem de tecnologias), ou seja, o PIM Designer informa a camada que ele deseja adaptar e quais as classes envolvidas. Os detalhes do que será adaptado será tratado posteriormente, também estando de acordo com as informações cadastradas na base de conhecimento.

A figura 28 mostra que as marcações do modelo devem estar de acordo com as informações previamente cadastradas na base de conhecimento. Do contrário não seria possível identificar as marcações.

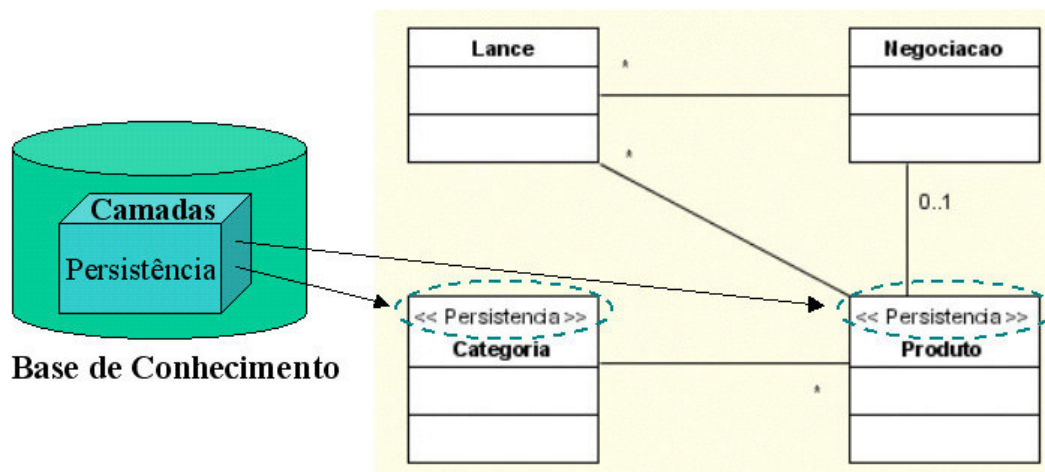


Figura 28 - Mapeamento Marcação (Modelo) x Camada (Base de Conhecimento)

4.3. Configuração da Base de Conhecimento

A base de conhecimento é um repositório que possui algumas informações necessárias para se fazer o mapeamento das transformações entre os modelos, usando scripts pré-cadastrados para diversos tipos de marcações. É nela que são configuradas as camadas a serem tratadas, as possíveis tags para cada camada e os arquivos RDLs correspondentes a cada tag. A figura 29 mostra a organização da base de conhecimento.

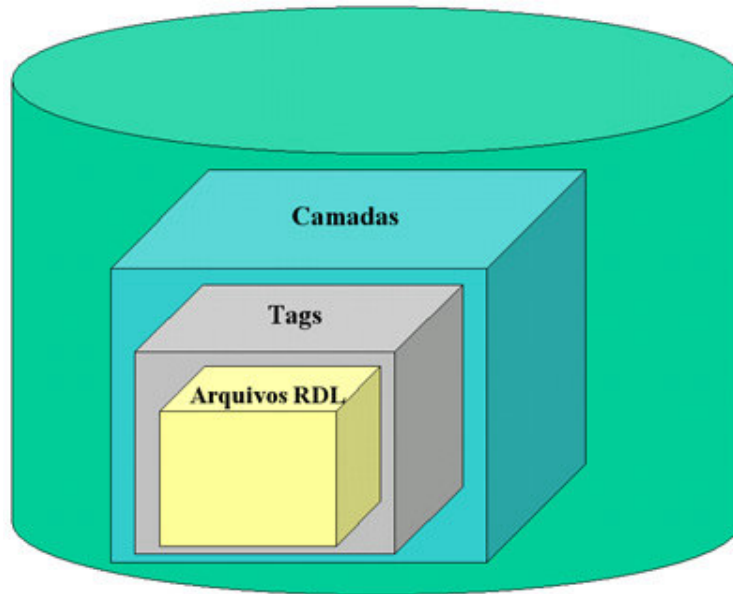


Figura 29 - Base de Conhecimento

As camadas são divisões lógicas dentro do modelo. Elas podem seguir padrões, como por exemplo, o MVC ou podem ser criadas através de critérios próprios. As marcações incluídas no modelo PIM, devem estar de acordo com as camadas cadastradas na base de conhecimento.

Cada camada pode possuir diversas tags, que identificam um framework, um pattern, entre outros tipos de tecnologias, que poderão ser utilizadas na geração do modelo PSM. Cada tag possui um ou mais arquivos RDL que contêm suas regras de transformações. Essas regras são o passo a passo das alterações que deverão ocorrer no modelo PIM, tendo como objetivo gerar o modelo PSM com as tecnologias adaptadas.

Na figura 30 podemos ver como a base de conhecimento é organizada, sendo possível visualizar as camadas, suas tags e os arquivos RDL representando as regras de transformações de cada tag.

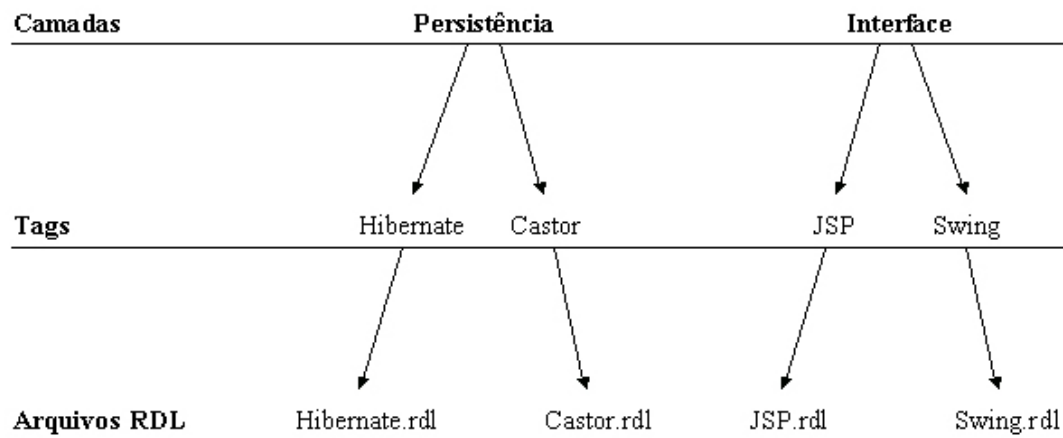


Figura 30 - Exemplificando a Base de Conhecimento

Logo, será necessário configurar a base de conhecimento, cadastrando as camadas que poderão ser tratadas durante o processo, as tags que cada camada poderá tratar e os arquivos RDL contendo as regras de transformações de cada tag.

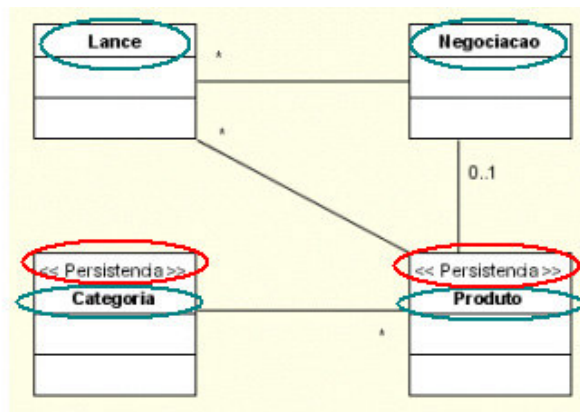
Conforme o processo for utilizado por repetidas vezes, e a configuração e a manutenção da base de conhecimento for sendo feita continuamente, a tendência é que a base de conhecimento se estabilize tornando essa etapa cada vez mais simples e menos trabalhosa.

Quanto maior o conhecimento dos Knowledge-Base Designers em relação a necessidade de adaptação dos modelos, maior poderá ser a preparação da base de conhecimento antes do processo ser iniciado. Com isso a chance de que a base de conhecimento tenha que ser alterada durante o processo diminui, agilizando ainda mais o processo como um todo.

4.4. Execução das Regras de Transformações

Para que as transformações possam ocorrer, é necessário que o modelo PIM esteja configurado com as marcações e que o mesmo esteja representado para o formato XMI. Somente assim será possível adquirir as informações do modelo necessárias ao processo proposto, e posteriormente atualizá-lo com novas informações.

A figura 31 a seguir mostra um modelo PIM e o seu arquivo XMI correspondente. Esse arquivo possui todas as informações do modelo, incluindo as classes, seus atributos, seus esteriótipos, entre outros detalhes. Uma parte das informações necessárias para que seja possível identificar que tipo de transformações devem ser feitas e onde as mesmas serão incluídas, estão contidas no arquivo XMI.



```

<UML:Class xmi.id = 'Ildb2215m108c0479313mm7f51' name = 'Lance' visibility = 'public'
  isSpecification = 'false' isRoot = 'false' isLeaf = 'false' isAbstract = 'false'
  isActive = 'false' />
<UML:Class xmi.id = 'Ildb2215m108c0479313mm7f3e' name = 'Negociacao' visibility = 'public'
  isSpecification = 'false' isRoot = 'false' isLeaf = 'false' isAbstract = 'false'
  isActive = 'false' />
<UML:Class xmi.id = 'Ildb2215m108c0479313mm7f2b' name = 'Produto' visibility = 'public'
  isSpecification = 'false' isRoot = 'false' isLeaf = 'false' isAbstract = 'false'
  isActive = 'false' />
<UML:ModelElement.stereotype>
  <UML:Stereotype xmi.idref = 'Ildb2215m108c0479313mm7e71' />
</UML:ModelElement.stereotype>
</UML:Class>
<UML:Class xmi.id = 'Ildb2215m108c0479313mm7f18' name = 'Categoria' visibility = 'public'
  isSpecification = 'false' isRoot = 'false' isLeaf = 'false' isAbstract = 'false'
  isActive = 'false' />
<UML:ModelElement.stereotype>
  <UML:Stereotype xmi.idref = 'Ildb2215m108c0479313mm7e71' />
</UML:ModelElement.stereotype>
</UML:Class>
<UML:Stereotype xmi.id = 'Ildb2215m108c0479313mm7e71' name = 'Persistencia'
  visibility = 'public' isSpecification = 'false' isRoot = 'false' isLeaf = 'false'
  isAbstract = 'false' />
<UML:Stereotype.baseClass>Class</UML:Stereotype.baseClass>
</UML:Stereotype>
    
```

Figura 31 - Modelo PIM e seu XMI correspondente

Esta fase está fortemente ligada às informações contidas na base de conhecimento, pois somente através dela será possível identificar as marcações utilizadas no modelo.

Esse mapeamento entre as marcações do modelo com as camadas da base de conhecimento é necessário, para que seja possível levantar as tags que poderão vir a ser adaptadas ao modelo. Fica a cargo do PSM Designer escolher aquelas que ele deseja que sejam utilizadas em conjunto com o modelo PIM, para gerar o modelo PSM.

A figura 32 abaixo mostra os Frameworks de Persistência, Hibernate e Castor, e os Patterns, DAO e Facade, como as possíveis Tags da camada de Persistência. Fica a cargo do PSM Designer selecionar aquelas que serão usadas para gerar o modelo PSM.

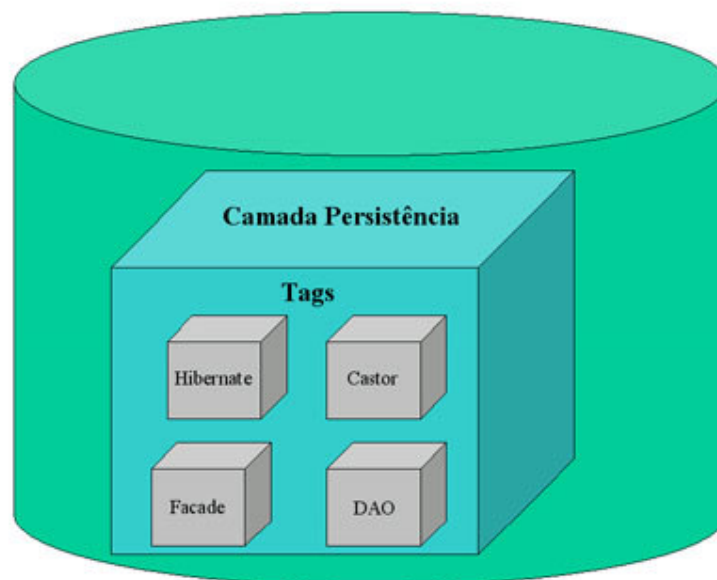


Figura 32 - Tags da Camada de Persistência

Após a escolha das tecnologias (Tags) pelo PSM Designer, a próxima etapa será a execução das regras de transformações das mesmas. Cada tecnologia está relacionada a um ou mais arquivos RDL que contêm o passo a passo da transformação a ser executada no modelo PIM de entrada.

A figura 33 mostra toda a estrutura da base de conhecimento, começando pela camada de Persistência, fornecida na configuração do modelo PIM (entrada), passando pela tag do framework Hibernate, escolhido pelo PSM Designer para gerar o modelo PSM (saída), e terminando com o arquivo Hibernate.rdl, que contém as regras necessárias para efetuar a transformação no modelo, gerando posteriormente o modelo PSM.

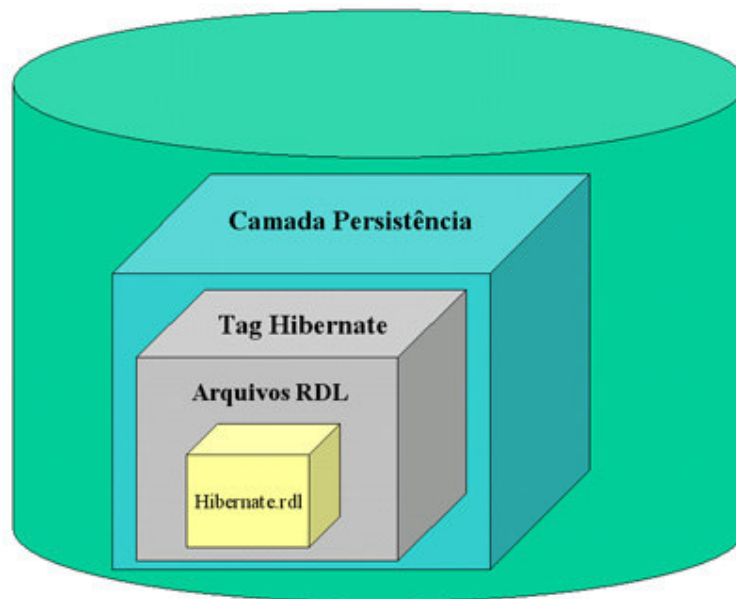


Figura 33 - Arquivos RDL da Tag Hibernate

As regras de transformações são enviadas para uma máquina virtual RDL, que foi desenvolvida durante a tese de doutorado do professor Dr. Toacy Cavalcante de Oliveira da Pontifícia Universidade Católica do Rio Grande do Sul, onde os arquivos RDL serão interpretados e o resultado atualizado em um arquivo XMI. Durante a execução desta etapa, será necessário que o PSM Designer informe alguns dados de acordo com a regra de transformação em questão, como por exemplo, nomes para novas classes.

Após a execução de todas as regras de transformações, o arquivo XMI irá conter as classes do PIM original, mais aquelas introduzidas pelo processo proposto.

Qualquer alteração que vise mudar o resultado de uma transformação deve ser feita nas regras contidas nos arquivos RDL. Dessa maneira as regras de transformações ficam independentes de qualquer outra parte do processo, facilitando assim a manutenção das mesmas.

Caso seja necessário criar uma nova classe e o nome da mesma só puder ser informado durante o processo de transformação, a linguagem RDL oferece esse recurso através do uso do argumento "?" (figura 34) como parâmetro para algumas operações. No momento em que a máquina virtual RDL encontra tal

parâmetro, ela pedirá ao PSM Designer que informe um nome para que o mesmo seja colocado na nova classe.

```
COOKBOOK Example
  RECIPE main
    NEW_METHOD (?,inserir);
    NEW_METHOD (?,alterar);
    NEW_METHOD (?,excluir);
    NEW_METHOD (SqlMapClient, startTransaction);
    NEW_METHOD (SqlMapClient, commitTransaction);
    NEW_METHOD (SqlMapClient, update);
  END_RECIPE;
END_COOKBOOK
```

Figura 34 - Arquivo RDL utilizando o caracter “?”

A figura 35 mostra parte do arquivo RDL Hibernate.rdl, que contém as regras de transformações do Framework Hibernate. Ele descreve a criação de duas classes, sendo que cada uma delas terá dois métodos. Serão incluídos também dois relacionamentos de dependência entre as classes marcadas no modelo e as que serão incluídas no mesmo.

Nesse caso o caracter “?” funciona de uma forma diferente. Pois quando é passado o nome das classes para a máquina virtual RDL, ela substitui automaticamente o caracter por esses nomes, não havendo a necessidade do PSM Designer inserir esses dados. As classes mapeadas no modelo PIM, foram passadas com esse objetivo.

```
COOKBOOK Example
  RECIPE main
    classe_session = NEW_CLASS (Session);
    classe_hibernateutil = NEW_CLASS (HibernateUtil);
    NEW_METHOD (classe_session, save);
    NEW_METHOD (classe_session, createQuery);
    NEW_METHOD (classe_hibernateutil, currentSession);
    NEW_METHOD (classe_hibernateutil, closeSession);
    LOOP
      classe_existente = ?;
      NEW_DEPENDENCY (classe_existente, classe_session);
      NEW_DEPENDENCY (classe_existente, classe_hibernateutil);
    LOOP;
  END_RECIPE;
END_COOKBOOK;
```


 **Classes Marcadas no Modelo**

Figura 35 - Exemplo de um Arquivo RDL

Durante a execução do arquivo RDL da figura 34, serão passadas as classes que foram marcadas no modelo, para que sejam inseridas no arquivo. A figura 36 mostra o modelo PSM, resultado da aplicação das transformações sobre o PIM da figura 28, acrescido das informações resultantes da execução do arquivo RDL da figura 34.

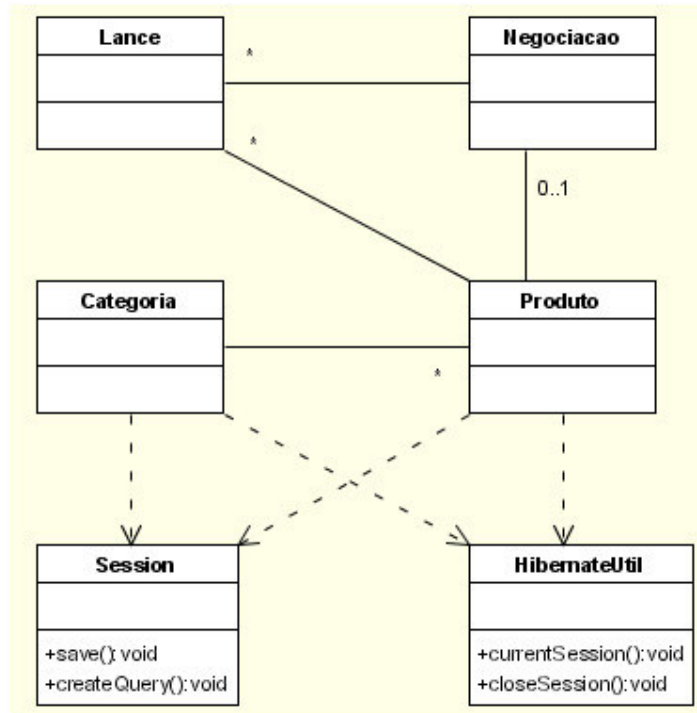


Figura 36 - Resultado da Transformação

4.5. Criação do Modelo PSM

A figura 37 mostra o modelo PSM gerado para o framework de persistência Castor. Esse é o modelo gerado a partir do modelo PIM inicial (figura 27), adicionadas as transformações escolhidas pelo PSM Designer, nesse caso o Framework de Persistência Castor. Resultando em um modelo específico para uma plataforma e contendo também as classes envolvidas na utilização do Framework em questão.

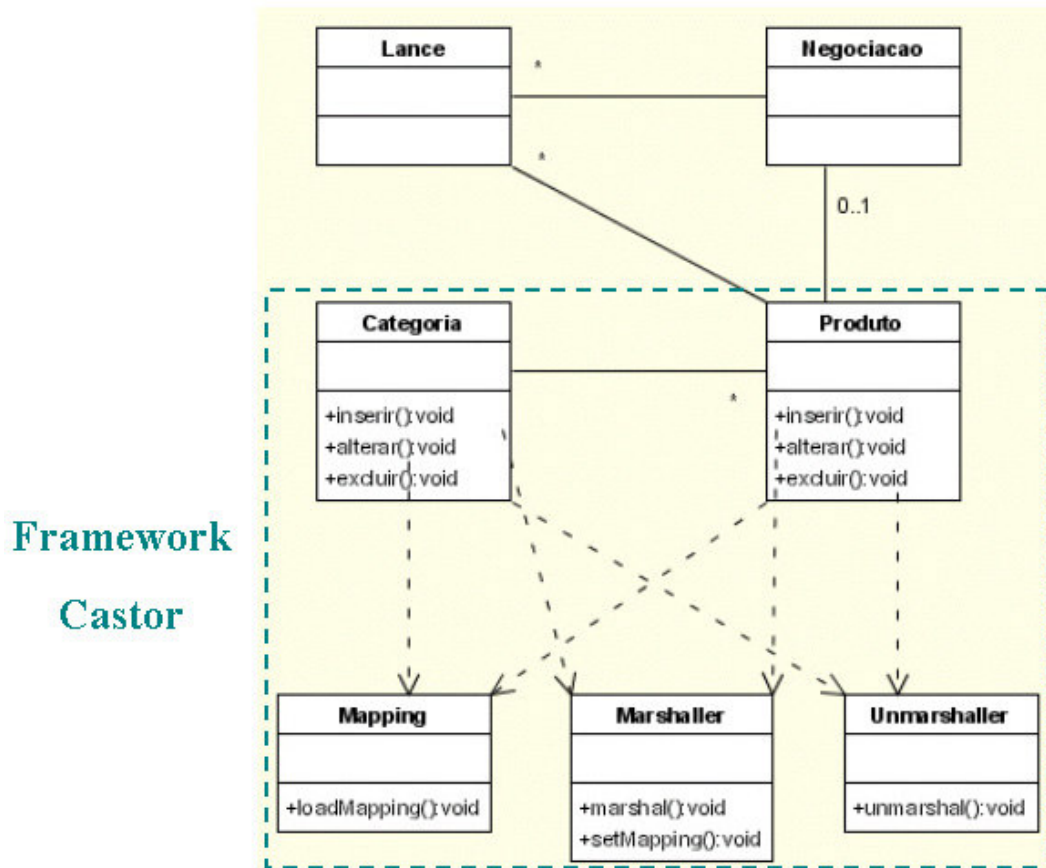


Figura 37 - Modelo PSM gerado para o Framework de Persistência Castor

O processo irá gerar também arquivos adicionais para a configuração e uso dos frameworks de persistência, tais como o XML de configuração de banco e o de mapeamento tabela-classe.

O arquivo XML da figura 38 representa a configuração do banco para o Framework Hibernate. Ele é gerado de maneira que o PSM Designer só precise preencher algumas informações, diminuindo o trabalho manual do mesmo.

```

<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
    "-//Hibernate/Hibernate Configuration DTD 2.0//EN"
    "http://hibernate.sourceforge.net/hibernate-configuration-2.0.dtd">

<hibernate-configuration>
  <session-factory name="java:comp/env/hibernate/SessionFactory">

    <property name="connection.datasource">datasource</property>
    <property name="dialect">net.sf.hibernate.dialect.MySQLDialect</property>
    <property name="show_sql">>false</property>
    <property name="use_outer_join">>true</property>
    <property name="transaction.factory_class">
      net.sf.hibernate.transaction.JTATransactionFactory
    </property>
    <property name="jta.UserTransaction">java:comp/UserTransaction</property>

    <mapping resource="org/hibernate/auction/Produto.hbm.xml"/>
  </session-factory>
</hibernate-configuration>

```

Figura 38 - Arquivo XML da configuração do banco para o Framework Hibernate

Abaixo a figura 39 ilustra o arquivo XML que representa o mapeamento tabela-classe, onde cada atributo da classe Produto é mapeado em um atributo da tabela Produto. Nesse arquivo também são descritas informações como chave, tipos, entre outras informações relativas ao mapeamento no banco.

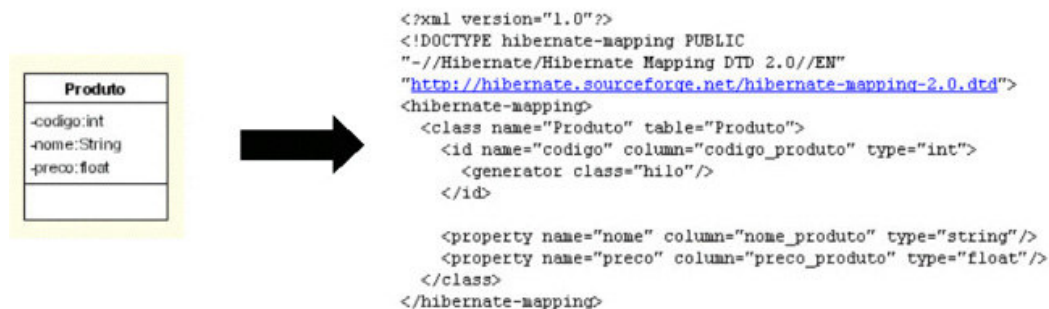


Figura 39 - Arquivo XML contendo o mapeamento tabela-classe do Framework Hibernate

4.6. Considerações Finais

O objetivo do processo é gerar um modelo para uma plataforma específica, de maneira a diminuir ao máximo o trabalho manual relacionado à inclusão de tecnologias no modelo.

Os limites envolvidos na abordagem proposta não levam em consideração a manutenção do código inserido após a geração do modelo PSM. Então durante a geração de um modelo PSM a partir de um modelo PIM, não há o tratamento do código existente no modelo PIM para ser adicionado ao modelo PSM gerado. Esse tipo de manutenção ficará a cargo do desenvolvedor, podendo o mesmo utilizar ferramentas que automatizem esse tipo de trabalho.

As alterações da estrutura dos modelos no código também não são tratadas pelo processo, havendo a necessidade do desenvolvedor efetuar as mudanças também nos modelos ou utilizar a própria ferramenta case para importar e atualizar essas alterações.

Utilizando o processo proposto, fica muito mais fácil e simples adaptar a arquitetura às novas necessidades em termos de tecnologias, pois o mesmo gera uma boa quantidade de informação que seria necessária que o PSM Designer inserisse manualmente. Desta forma será possível utilizar tecnologias distintas de persistência com um custo relativamente baixo de adaptação.