5
O Editor de Diagramas

O objetivo do editor de diagramas dirigido por metamodelos é trazer ao usuário a flexibilidade de se trabalhar com várias linguagens de modelagem de IHC em um ambiente único, no qual é possível definir elementos e regras de sintaxe a partir de um arquivo XML sem necessidade de recompilação do editor.

A ferramenta desenvolvida dá ao usuário a possibilidade de criação de seu próprio metamodelo, incluindo símbolos e regras sintáticas. Esse metamodelo é importado pelo editor de diagramas, que passa por uma validação de sintaxe para estar pronto para a utilização. Com isso, ao importar o metamodelo criado, o editor se tornará um editor específico da linguagem, sensível à sua sintaxe. Com o seu detector dinâmico de eventos, cada passo do usuário é avaliado e caso ele cometa um erro sintático ele é alertado.

5.1
Organização do metamodelo

Ao se criar um metamodelo para o editor, deve-se seguir a seguinte organização de arquivos exemplificada na Figura 47. O nome da pasta é o nome do metamodelo e os nós utilizados no metamodelo devem ser criados na ferramenta Dia [Dia, 2011]. Nela, os elementos devem ser exportados no formato SHAPE. A cada nó exportado são gerados dois arquivos: o arquivo .SHAPE, que é um arquivo vetorial semelhante ao SVG, e um arquivo .PNG, que é uma imagem referente ao .SHAPE, criado para ser o ícone de representação na biblioteca de elementos. Além desses dois tipos de arquivo, temos um terceiro arquivo, que é o metamodel.xml que guarda todos os dados referentes ao metamodelo. Ele tem informações sobre a lista de nós e arestas pertencentes ao metamodelo, além de todas as suas respectivas regras. As arestas são criadas dentro do arquivo metamodel.xml e não necessita do aplicativo Dia. Mais informações serão dadas na seção que descreve o metamodel.xml.

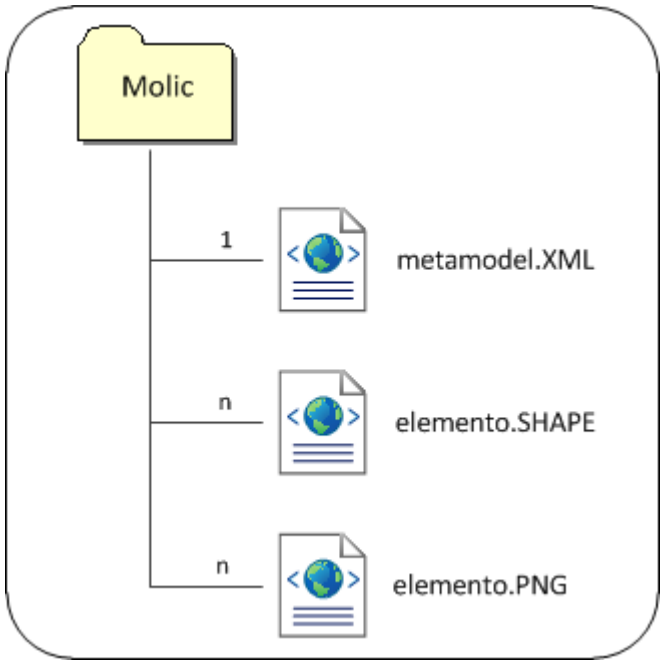

Figura 47 - Organização da pasta do metamodelo.

Além da criação dos arquivos SHAPE, figuras do tipo PNG e JPEG também são aceitas para representação dos elementos. Porém, a perda de qualidade é perceptível ao se redimensionar um elemento, o que não acontece com o formato SHAPE, que é baseado em gráficos vetorizados SVG [SVG, 2010]. Nestes casos, a mesma figura que representa o ícone do elemento na biblioteca é utilizado na sua representação no modelo.

5.2
Descrição do metamodelo

A linguagem XML tem se mostrado adequada e se tornado padrão para modelagem de diversos tipos de arquivo por apresentar características como flexibilidade e facilidade de aprendizado [Almeida, 2002]. Por isso, ela foi utilizada no trabalho para estruturar a linguagem do metamodelo. Esse arquivo deve seguir a formatação definida pelo XML Schema [XML Schema, 2010] apresentado no Apêndice C.

O arquivo XML é o responsável por guardar as informações necessárias para a composição do metamodelo. Nele é encontrada a listagem de todos os nós existentes na pasta e a listagem de todas as arestas com as suas devidas definições. As arestas não necessitam de arquivos gráficos para serem representadas porque elas são definidas no próprio metamodelo. Mais

informações sobre a definição das arestas estão disponíveis no capítulo do metamodelo.

Como apresentado na Figura 48, a formatação do arquivo metamodel.xml se divide da seguinte forma: em primeiro lugar vem os elementos do metamodelo que se dividem em nós e arestas; em segundo lugar vem as regras do metamodelo que definem como os elementos definidos se relacionam.

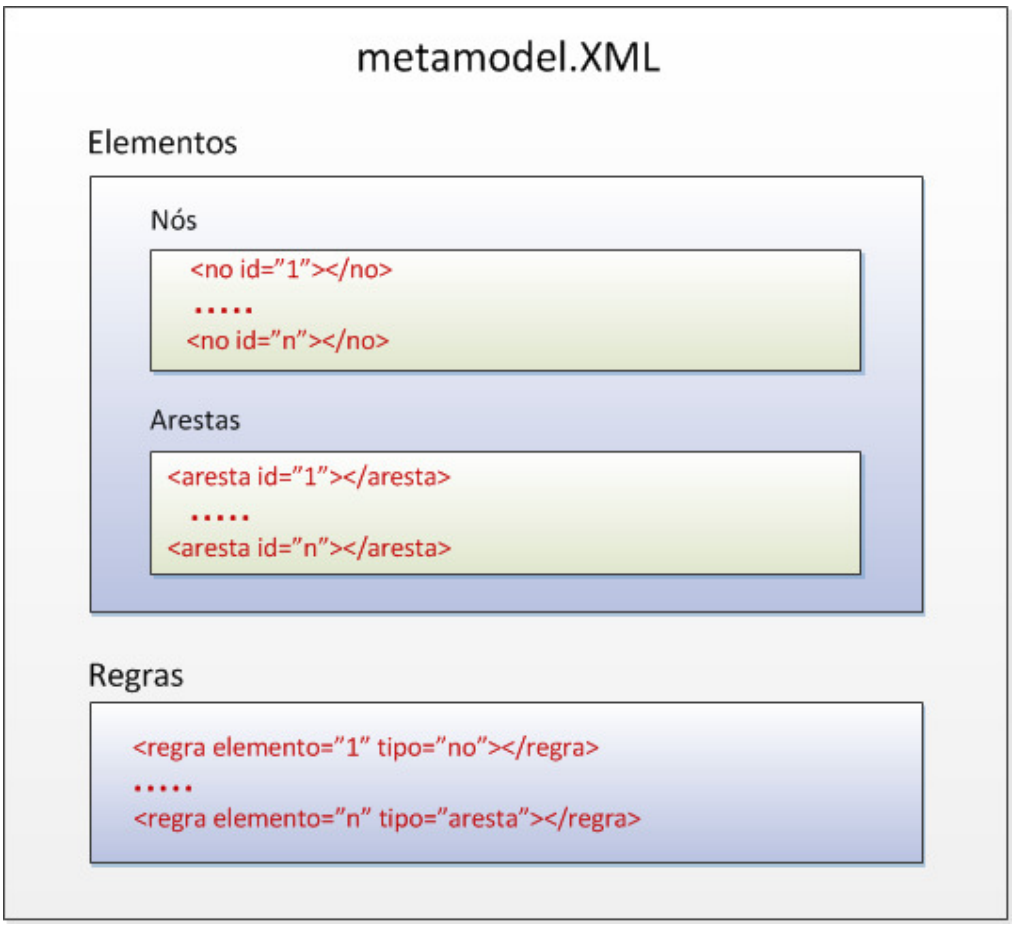

Figura 48 - Organização do arquivo do metamodelo.

5.3
O aplicativo

O editor de diagramas é um aplicativo construído inteiramente em JAVA. Como exemplificado na Figura 49, a estrutura do editor se divide em três blocos: JgraphX, metamodelo e XML. A biblioteca JgraphX foi modificada para atender as especificações do trabalho Ela auxilia na manipulação de objetos relacionados a grafo. Funções básicas de um editor como: arraste de objetos,

ligação com arestas, copiar e colar já são oferecidas prontas pela biblioteca. Na parte do metamodelo, foram implementadas as classes que controlam os nós e arestas e seus respectivos atributos, a classe que controla as regras do metamodelo e a classe que valida a sintaxe do modelo elaborado pelo usuário. Por ultimo, a classe que lê o arquivo metamodel.xml e repassa todos os dados a classe Metamodelo.

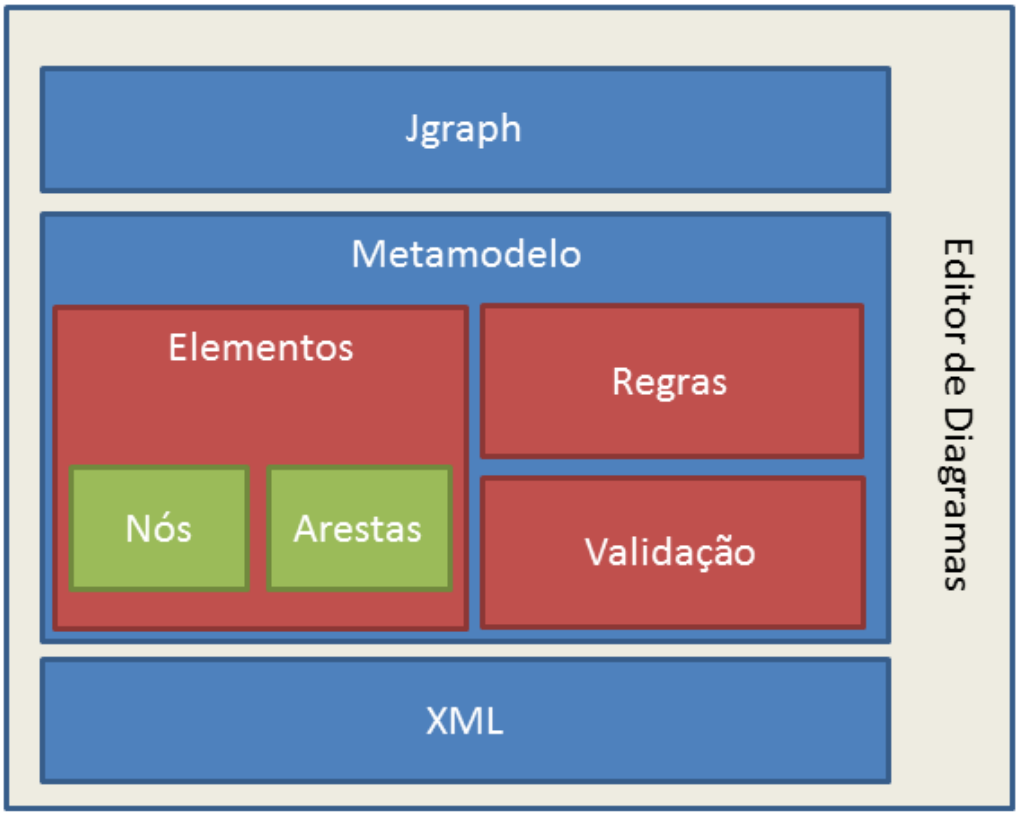

Figura 49 - Arquitetura do editor de diagramas.

Ao executar o editor de diagramas, a primeira coisa a se fazer é carregar o metamodelo desejado pela pasta, através da opção *File/Import Metamodel*. Nesse momento, o arquivo metamodel.xml é lido e transferido para a classe do metamodelo que verifica a corretude dos dados. Caso algum dado esteja incorreto, um alerta aparecerá indicando o erro para o usuário.

Caso não exista nenhum problema na importação, os metamodelos do metamodelo serão apresentados na janela à esquerda do aplicativo, exemplificado na figura abaixo. Na janela abaixo dos elementos do metamodelo

existe um visualizador da folha de trabalho completa onde é possível controlar a ampliação da janela direita da aplicação onde se elabora a solução (Figura 50).

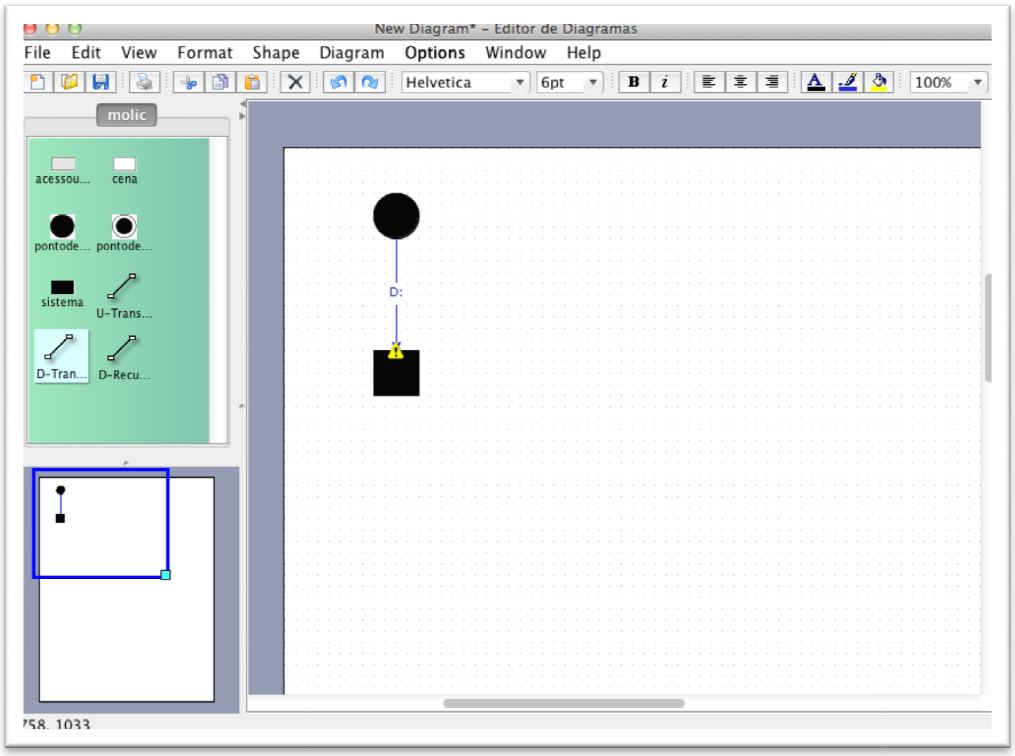

Figura 50 - Tela do editor de diagramas.

Ao se cometer alguma infração às regras estipuladas da linguagem, um ícone de alerta irá aparecer no elemento. Colocando o mouse por cima do ícone, uma mensagem de erro aparecerá indicando a causa do erro. A Figura 51 exemplifica esse evento.

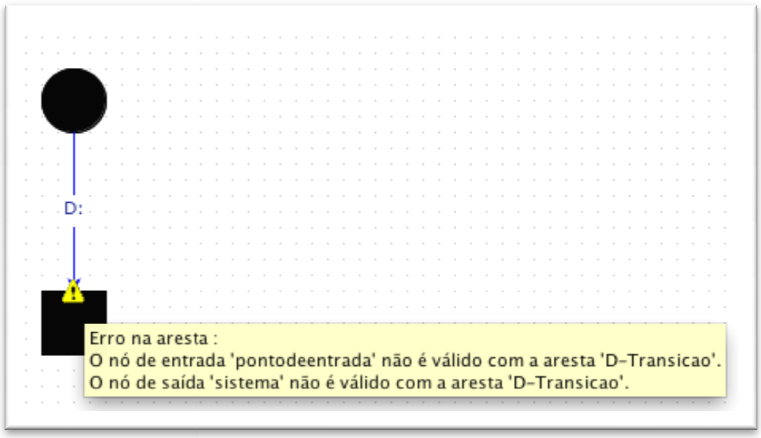

Figura 51 - Alerta de erro de sintaxe.