Aldo Ianelo Guerra
Victor Aoqui

Plataforma de Simulação de Grades Computacionais: Interface Icônica e Interpretador de Modelos Externos

Monografia apresentada ao Departamento de Ciências de Computação e Estatística do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", como parte dos requisitos necessários para aprovação na disciplina Projeto Final.

São José do Rio Preto
2010

Aldo Ianelo Guerra
Victor Aoqui

Plataforma de Simulação de Grades Computacionais: Interface Icônica e Interpretador de Modelos Externos

Monografia apresentada ao Departamento de Ciências de Computação e Estatística do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", como parte dos requisitos necessários para aprovação na disciplina Projeto Final.

Orientadora:
Profa. Dra. Renata Spolon Lobato
Co-orientador:
Prof. Dr. Aleardo Manacero Júnior

São José do Rio Preto
2010

Aldo Ianelo Guerra
Victor Aoqui

Plataforma de Simulação de Grades Computacionais: Interface Icônica e Interpretador de Modelos Externos

Monografia apresentada ao Departamento de Ciências de Computação e Estatística do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", como parte dos requisitos necessários para aprovação na disciplina Projeto Final.

Profa. Dra. Renata Spolon Lobato Aldo Ianelo Guerra Victor Aoqui

Banca Examinadora:
Prof. Dr. José Márcio Machado
Prof. Dr. Masayoshi Tsuchida

São José do Rio Preto
2010

Dedicatória

Aldo:
Aos meus pais, Adevaldo e Alzira.
À minha irmã, Jaqueline.

Victor:
Aos meus pais, Carlos e Fumiko.
À minha irmã, Vanessa.

Agradecimentos

Gostaria de agradecer aos meus pais, Adevaldo e Alzira, e a minha irmã Jaqueline que me apoiaram em todos os momentos que precisei e em todas as decisões que tomei.

Agradeço também aos meus tios e tias que sempre me trataram como um filho e me auxiliaram em diversos momentos.

Aos amigos que conheci durante a faculdade, Marco Antonio, Paulo Henrique e Tiago e se tornaram meus grandes amigos.

A todos os membros do Grupo de Sistemas Paralelos e Distribuídos que me auxiliaram sempre que precisei.

À minha orientadora, Profa. Dra. Renata Spolon Lobato, e ao meu orientador, Prof. Dr. Aleardo Manacero Júnior, pela orientação, dedicação e profissionalismo de suas partes.

À FAPESP, que me auxiliou financeiramente com uma bolsa de iniciação científica (processo nº 2009/00502-0) para o desenvolvimento deste projeto e também concedeu uma bolsa de auxílio pesquisa (processo nº 2008/09312-7) para a compra dos computadores do Grupo de Sistemas Paralelos e Distribuídos.

Aos demais professores do IBILCE/UNESP, e a todas as pessoas que também contribuíram com seu conhecimento e dedicação para minha formação acadêmica e profissional.

Aldo Ianelo Guerra.

Agradecimentos

Em primeiro lugar, gostaria de agradecer aos meus pais, Carlos e Fumiko, pelo apoio que recebo diariamente, tanto nos momentos felizes quanto nos momentos díficeis. Eles são imprescindíveis em minha vida.

Agradeço também à minha irmã, Vanessa, uma pessoa dedicada, determinada e que sempre me auxilia quando necessário.

Aos meus primos, Eduardo, Fábio, Fernando e Gerson, que são como irmãos para mim, companheiros para todas as horas. Muito obrigado primos!

À minha orientadora, Profa. Dra. Renata Spolon Lobato, e ao meu orientador, Prof. Dr. Aleardo Manacero Júnior, pela orientação, dedicação e profissionalismo de suas partes.

À FAPESP, que me auxiliou financeiramente com uma bolsa de iniciação científica (processo nº 2009/00160-2) para o desenvolvimento deste projeto e também concedeu uma bolsa de auxílio pesquisa (processo nº 2008/09312-7) para a compra dos computadores do Grupo de Sistemas Paralelos e Distribuídos.

Aos demais professores do IBILCE/UNESP que também contribuíram com seu conhecimento e dedicação para minha formação acadêmica e profissional.

Aos integrantes do Grupo de Sistemas Paralelos e Distribuídos, que me auxiliaram no desenvolvimento deste projeto.

Aos meus amigos que conheci na faculdade, Antonio, Honda, Paraná, Rafael Braga, Rafael Yamato, grandes amigos que jamais serão esquecidos.

Victor Aoqui.

Resumo

Este trabalho está inserido no contexto de um projeto de construção de uma plataforma de simulação de grades computacionais e seu objetivo é apresentar as gramáticas especificadas para as linguagens dos modelos aceitos pelo simulador, um protótipo para a interface icônica do simulador de grades computacionais e um protótipo para o interpretador de modelos SimGrid. O protótipo deste simulador possibilita ao usuário construir modelos de simulação através de uma interface icônica intuitiva e de fácil utilização e também, importar modelos do simulador SimGrid. Salienta-se que o simulador foi totalmente implementado em linguagem Java, pois dessa forma faz com que a ferramenta tenha um alto nível de portabilidade. Além disso, foi utilizada a ferramenta Java Compiler Compiler 5.0 (JavaCC) para auxiliar na construção do interpretador de modelos SimGrid. Para a compilação dos códigos-fontes foi utilizado o compilador Java Development Kit (JDK) 1.6.0 build 21 em conjunto com o Apache Ant 1.8.1, que possibilitou a automatização do processo de compilação de todos os códigos-fontes e da geração do arquivo executável jar para o simulador. Os testes da ferramenta foram realizados nos sistemas operacionais GNU/Linux Debian Lenny e Microsoft Windows.

Palavras-chave: simulação. grades computacionais. gramáticas. linguagens. SimGrid. interface icônica.

Abstract

This work is placed in the context of a project to build a simulation plataform for grid computing and its purpose is to present the grammars specified for the languages of the models accepted by the simulator, a prototype for an iconic interface for the grid computing simulator and a prototype for the interpreter of SimGrid models. The prototype of this simulator allows the user to build simulation models through an intuitive and of easy manipulation iconic interface and also import SimGrid simulator models. It is noted that the simulator has been fully implemented in Java language, because then it makes the tool to have a high level of portability. Besides that, the Java Compiler Compiler 5.0 (JavaCC) was used to assist in building the SimGrid models interpreter and the internal models interpreters of the simulator of this project. The Java Development Kit 1.6.0 build 21 compiler was used in conjunction with Apache Ant 1.8.1 to compile the source codes, which enabled the automation of the process of compiling all the source codes and generating the executable jar file for the simulator. The tests of the tool were performed in the GNU/Linux Debian Lenny and Microsoft Windows operating systems.

Keywords: simulation. grid computing. grammars. languages. SimGrid. iconic interface.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

AMOK: *Advanced Metacomputing Overlay Kit*
API: *Application Programming Interface*
BNF: *Backus-Naur Form*
CS: Centro de Serviço
DAG: *Direct Acyclic Grafs*
DB: *Database*
FPLTF: *Fastest Processor to Largest Task First*
GSPD: Grupo de Sistemas Paralelos e Distribuídos
GNU: *General Public License*
GRAS: *Grid Reality And Simulation*
ID: Identificador
JavaCC: *Java Compiler Compiler*
JDK: *Java Development Kit*
JVM: *Java Virtual Machine*
MPI: *Message Passing Interface*
MSG: Meta-SimGrid
UCP: Unidade Central de Processamento
RR: *Round-Robin*
SG: SimGrid
SMPI: SimGrid *Message Passing Interface*
SPEC: *Standard Performance Evaluation Corporation*
UML: *Unified Modeling Language*
WIMP: *Windows, Icons, Menus and Pointers*
XBT: *Extended Bundle of Tools*
XML: *eXtensible Markup Language*

Capítulo 1 - Introdução

1.1 Considerações Iniciais

Os avanços tecnológicos dos microprocessadores e das redes de comunicação possibilitaram diversas mudanças. Dentre elas destaca-se a evolução da computação seqüencial e centralizada para a computação baseada em recursos distribuídos geograficamente, caracterizando os chamados sistemas distribuídos [Coulouris *et al.*, (2005)].

Sistemas distribuídos são constituídos de computadores independentes interligados por redes de computadores e que utilizam um *software* apropriado, denominado *middleware*, para gerenciar suas tarefas [Coulouris *et al.*, (2005)]. Como exemplo destes sistemas, pode-se citar as grades computacionais, utilizadas amplamente nas aplicações científicas e comerciais.

Grades computacionais possibilitam o compartilhamento de recursos como, por exemplo, processamento, armazenamento e serviços. No entanto, apresentam uma série de problemas, tais como a sincronização de dados, segurança na utilização do sistema e comunicação limitada. Estes problemas devem-se principalmente ao

alto grau de distribuição geográfica dos recursos e à grande heterogeneidade de domínios, máquinas e sistemas operacionais pertencentes à grade computacional [Reis, (2005)].

1.2 Motivação e Escopo

A utilização de grades computacionais tem aumentado gradativamente ao longo dos anos, o que gera uma demanda por ferramentas adequadas que simulam e avaliam o desempenho destes sistemas. Para isso, são utilizados modelos de simulação. Em geral, as ferramentas de simulação disponíveis atualmente não são simples de serem utilizadas e geram um modelo exclusivo para sua própria ferramenta, ou seja, não permitem a simulação de modelos gerados por outros simuladores. Todavia, nenhuma delas possui interface amigável para o usuário.

A fim de corrigir tais deficiências, este projeto está inserido no contexto do desenvolvimento de uma plataforma de simulador de grades computacionais inovador. O usuário poderá criar um modelo simulável específico do simulador proposto ou ainda utilizar modelos gerados por simuladores externos para realizar a simulação. No término da simulação a ferramenta disponibilizará dados estatísticos da simulação ao usuário. Além disso, esta ferramenta será de fácil utilização, visto que será o primeiro simulador de grades computacionais a possuir interface icônica. Através dessa plataforma, o usuário criará um modelo icônico da grade computacional, que será posteriormente convertido para um modelo simulável através de um interpretador. Este modelo simulável, escrito na forma de *script*, será simulado para disponibilizar métricas de desempenho que sejam interessantes para o usuário.

Este projeto abrange a parte da especificação de uma gramática para o modelo utilizado pelo motor de simulação, uma gramática para o modelo icônico e outra para conversão de modelos SimGrid [Casanova, (2001)], além da especificação, construção e testes dos protótipos para a interface icônica do simulador e para os interpretadores de modelos simuláveis, de modelos icônicos e de modelos SimGrid.

1.3 Objetivo e Metodologia

Este trabalho está inserido no contexto do desenvolvimento de uma plataforma de simulador de grades computacionais inovador. O projeto do simulador envolve os problemas de especificação de linguagens, a construção dos interpretadores destas linguagens, a especificação e implementação de uma interface icônica e, por fim, a implementação de um motor de simulação capaz de prover medidas de desempenho e permitir mudanças nos modelos simulados em tempo de execução. No entanto, neste projeto serão abordados a construção de um protótipo para a interface icônica do simulador e os protótipos dos interpretadores de modelos simuláveis, de modelos icônicos e de modelos SimGrid.

O usuário poderá criar um modelo simulável específico do simulador proposto ou ainda utilizar modelos gerados por simuladores externos para realizar a simulação. No término da simulação a ferramenta disponibilizará dados estatísticos da simulação ao usuário. Além disso, esta ferramenta será de fácil utilização, visto que será o primeiro simulador de grades computacionais a possuir interface icônica.

Na Figura 1.1 é apresentada conceitualmente a plataforma proposta. Através dessa plataforma, o usuário criará um modelo icônico da grade computacional, que será posteriormente convertida para um modelo simulável através de um interpretador. Este modelo simulável, escrito na forma de *script*, será simulado para disponibilizar métricas de desempenho que sejam interessantes para o usuário.

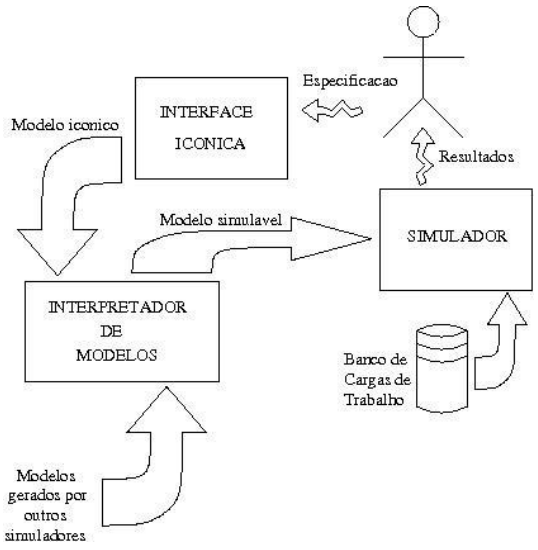

Figura 1.1: Diagrama conceitual da plataforma de simulação.

Salienta-se que este trabalho abrange as seguintes atividades:

- Especificação de uma linguagem e sua respectiva gramática para o modelo simulável;
- Especificação de uma linguagem e sua respectiva gramática para o modelo do sistema;
- Especificação da gramática da linguagem de modelos SimGrid;
- Especificação, implementação e testes de um protótipo para o interpretador de modelos externos (SimGrid);
- Especificação, implementação e testes de protótipos para os interpretadores de modelos internos (modelo simulável e modelo do sistema);
- Especificação, implementação e testes de um protótipo para a interface icônica do simulador.

Já o motor de simulação e o interpretador do modelo simulável estão sendo especificados e construídos por outros membros do Grupo de Sistemas Paralelos e Distribuídos (GSPD) [Oliveira *et al*., (2009)] [Oliveira *et al*., (2010)].

A metodologia adotada para o desenvolvimento do projeto é descrita nas etapas a seguir:

1. Estudo detalhado dos simuladores existentes, conduzido por Victor e Aldo;
2. Levantamento dos requisitos funcionais e de métricas de desempenho necessárias para um ambiente de simulação, conduzido por Victor e Aldo;
3. Especificação da linguagem gráfica para o simulador, conduzida por Aldo;
4. Especificação da linguagem utilizada pelo modelo simulável (utilizado no motor de simulação) para o simulador, conduzida por Aldo e Victor;
5. Especificação de uma gramática para a linguagem icônica, conduzida por Aldo;
6. Especificação de uma gramática para a linguagem utilizada pelo modelo simulável, conduzida por Aldo e Victor;
7. Especificação da gramática para a linguagem de modelos SimGrid, conduzida por Victor;

8. Implementação dos analisadores léxico, sintático e semântico para a linguagem icônica, conduzida por Aldo;
9. Implementação dos analisadores léxico, sintático e semântico para a linguagem do modelo simulável, conduzida por Aldo;
10. Implementação dos analisadores léxico, sintático e semântico para a linguagem de modelos SimGrid, conduzida por Victor;
11. Implementação do protótipo da interface icônica do simulador, conduzida por Aldo;
12. Implementação do protótipo dos interpretadores de modelos internos (modelo simulável e modelo do sistema), conduzida por Aldo;
13. Implementação do protótipo do interpretador de modelos SimGrid, conduzida por Victor;
14. Testes do protótipo da interface do simulador, conduzido por Aldo;
15. Testes do protótipo dos interpretadores de modelos internos, conduzido por Aldo;
16. Testes do protótipo do interpretador de modelos SimGrid, conduzido por Victor;
17. Redação da monografia, conduzida por Aldo e Victor;
18. Apresentação da monografia, conduzida por Aldo e Victor.

Este projeto foi desenvolvido em ambiente GNU/Linux Debian Lenny e Microsoft Windows. A linguagem de programação utilizada para implementar todo o sistema foi a linguagem Java [Java, (2010)] [Deitel e Deitel, (2006)] e o compilador utilizado é o *Java Development Kit* (JDK) 1.6.0 *build* 21 [JDK, (2010)] em conjunto com o Apache Ant 1.8.1 para automatizar o processo de compilação dos códigos-fontes e da geração do arquivo executável jar para o simulador.

O simulador SimGrid será utilizado para realizar simulações no *cluster Beowulf* financiado pela FAPESP (processo nº 2004/01340-0) do laboratório do GSPD.

Além disso, foi utilizada a ferramenta Java Compiler Compiler (JavaCC) 5.0 [JavaCC, (2010)] [Delamaro, (2004)], totalmente compatível com a linguagem de programação Java, para auxiliar na construção do interpretador de modelos externos e dos analisadores léxico e sintático para as linguagens citadas acima.

1.4 Organização da Monografia

Esta monografia está organizada da seguinte maneira:

- Capítulo 2: apresenta-se uma revisão bibliográfica dos assuntos relacionados com o projeto;
- Capítulo 3: apresentam-se as especificações das linguagens utilizadas pelo protótipo do simulador, a especificação e construção do protótipo do interpretador de modelos SimGrid, a especificação e construção do protótipo do interpretador de modelos icônicos, a especificação e construção do protótipo do interpretador de modelos simuláveis e a especificação e implementação da interface icônica;
- Capítulo 4: apresentam-se os testes realizados com o protótipo do interpretador de modelos SimGrid, com o protótipo do interpretador de modelos icônicos, com o protótipo de modelos simuláveis e com o protótipo da interface icônica, assim como os resultados obtidos;
- Capítulo 5: contribuições, dificuldades encontradas, conclusões e direcionamento para trabalhos futuros.

Capítulo 2 - Revisão Bibliográfica

2.1 Considerações Iniciais

Este capítulo aborda os temas relacionados com o desenvolvimento do projeto do simulador de grades computacionais.

Na seção 2.2 será feita uma revisão dos principais simuladores de grades computacionais existentes. A seção 2.3 apresenta a modelagem de simulações através de redes de filas. A seção 2.4 discute princípios, técnicas e conceitos para projeto de interface icônica. Por fim, a seção 2.5 faz algumas considerações finais a respeito deste capítulo.

Este estudo visa à estruturação de uma base sólida para permitir a identificação dos parâmetros de entrada e saída necessários para um simulador de grades computacionais e a especificação da interface icônica e de linguagens e gramáticas para os modelos internos (simulável e icônico) e modelos SimGrid. Com isso, implementa-se a interface icônica e os interpretadores dessas linguagens.

Nas seções seguintes são detalhados os estudos realizados a respeito de cada assunto abordado.

2.2 Simuladores de Grades Computacionais

A simulação de grades computacionais vem sendo estudada em razão da dificuldade da implementação de plataformas reais para o estudo das mesmas. Aspectos da implementação de uma grade computacional como, por exemplo, o poder computacional necessário, podem ser estudados antecipadamente com o uso de simuladores.

Nesta seção são relatados os estudos dos simuladores SimGrid [Casanova, (2001)], OptorSim [Bell *et al.*, (2003)], Bricks [Takefusa *et al.*, (1999)] e GridSim [Sulistio *et al.*, (2008)] e suas contribuições para o projeto.

O foco dos estudos realizados foi embasado no simulador SimGrid, já utilizado dentro do GSPD do Instituto de Biociências, Letras e Ciências Exatas - Universidade Estadual Paulista "Júlio de Mesquita Filho" (IBILCE/UNESP). Este estudo foi importante principalmente para adiquirir mais conhecimento em algoritmos de escalonamento [Falavinha Jr., (2006)(1)] [Falavinha Jr., (2006)(2)]. A importância deste simulador consiste no fato de que a modelagem da grade computacional é realizada separadamente, possibilitando abstrair os requisitos necessários de um simulador.

O estudo do simulador Bricks é importante para o projeto, visto que ele é implementado através da linguagem Java e utiliza redes de filas para efetuar a simulação. Já o OptorSim foi inserido no estudo em razão de efetuar a simulação por meio de *sites*, evidenciando um método de simulação diferenciado baseado em um projeto real. Por último, o GridSim foi estudado a fim de obter mais informações relevantes a serem consideradas durante o projeto de um simulador de grades computacionais.

2.2.1 SimGrid

O SimGrid foi criado em 1999 por Henri Casanova para que fosse possível o estudo de algoritmos de escalonamento sem a necessidade de se implementar um sistema real para isso. Inicialmente, o SimGrid consistia em uma ferramenta de simulação orientada a eventos, permitindo construir simulações de aplicações

específicas. O simulador implementava uma interface, ou API (*Application Programming Interface*), escrita em linguagem C, denominada SG. Com tal API, era possível especificar a simulação do escalonamento de tarefas em determinados recursos.

Na versão atual (versão 3) são implementados diferentes ambientes de programação que funcionam sobre o mesmo núcleo de simulação. Isso provê versatilidade ao simulador, já que cada ambiente constitui um paradigma diferente, sendo destinado a usuários específicos. Na Figura 2.1 são apresentados os componentes do SimGrid.

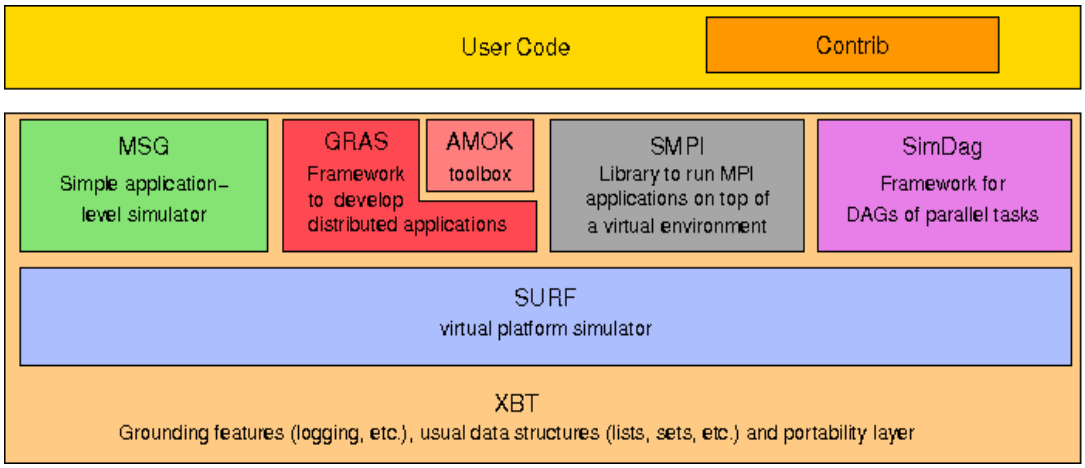

Figura 2.1: Componentes do SimGrid [SimGrid, (1999) (1)].

Módulos do SimGrid

***Meta-SimGrid* (MSG) [MSG, (1999)]**

O MSG foi o primeiro ambiente de programação distribuído desenvolvido no SimGrid visando a simulação em termos dos agentes de comunicação, portanto o realismo da simulação não é o principal objetivo deste módulo, sendo que muitos detalhes técnicos da plataforma do *grid* são omitidos.

Construir um programa de simulação usando o MSG envolve a codificação de cada agente e de cada recurso. Um agente representa uma entidade que toma as decisões de escalonamento, ou seja, interage enviando, recebendo ou processando

uma tarefa executada em uma determinada localidade. Uma tarefa é uma atividade, a qual pode ser uma computação ou uma transferência de dados.

***Grid Reality and Simulation* (GRAS) [GRAS, (1999)]**

O GRAS provê a implementação de aplicações distribuídas em plataformas heterogêneas. O ambiente GRAS deve ser usado se o interesse for produzir um programa que possa ser usado em um sistema distribuído [GRAS, (1999)]. Isso traz facilidade de uso e de controle do simulador durante o desenvolvimento.

Juntamente como o GRAS, existe um *toolkit* chamado *Advanced Metacomputing Overlay Kit* (AMOK) que implementa em alto nível diversos serviços necessários para várias aplicações distribuídas.

SMPI [SMPI, (1999)]

O ambiente SMPI permite a simulação de programas *Message Passing Interface* (MPI) já prontos, ou seja, nesse ambiente pode-se simular MPI sem que seja necessária qualquer modificação [SMPI, (1999)]. O ambiente SMPI ainda está em desenvolvimento.

SimDag [SimDag, (1999)]

O SimDag fornece funcionalidades para simular tarefas de programação no modelo DAG (*Direct Acyclic Graf*s) [SimDag, (1999)], ou seja, é possível especificar relações de dependências entre tarefas de um programa paralelo.

SURF [SURF, (1999)]

O SURF fornece as principais ferramentas para executar a simulação, serve de base para execução da simulação dos níveis acima dele, ou seja, ele é o núcleo de todos os módulos utilizados no SimGrid [SURF, (1999)]. Existente na versão 3 do SimGrid, substituiu o SG [SimGrid, (1999)(2)], que era implementado na primeira versão do simulador.

Extended Bundle of Tools (XBT)

O XBT representa o núcleo de ferramentas SimGrid implementando diversas funcionalidades que auxiliam o SURF como, por exemplo, estruturas de dados, serviço de *logging* e manipulação de grafos [Oliveira, (2008)].

Um importante recurso do SimGrid consiste em especificar a plataforma e os processos utilizados fora do arquivo de simulação. Essas especificações são feitas em arquivos XML (*Extensible Markup Language*). Isso proporciona flexibilidade na hora de testar um mesmo algoritmo em diferentes plataformas.

Exemplo de Simulação Usando SimGrid

Neste tópico será demonstrado como se efetua uma simulação através do SimGrid usando o módulo MSG. Foi simulado um pequeno *cluster* de nove nós utilizando o algoritmo *round-robin* para o escalonamento de 20 tarefas atribuídas ao sistema para avaliar o processo de simulação e os resultados obtidos.

A simulação através do SimGrid ocorre pela elaboração de 3 arquivos. Um arquivo contém a especificação da plataforma a ser utilizada, o segundo contém as especificações sobre a aplicação a ser simulada e o terceiro arquivo é um executável compilado em C usando as APIs do SimGrid.

Na Figura 2.2 é apresentado o arquivo plataform_file. Este arquivo contém a especificação da plataforma e nele são apresentados os seguintes itens:

- Poder computacional de cada máquina em Mflops/s;
- Links de conexão que serão usados para conectar os nós do sistema, a banda de cada conexão em Mb/s e suas latências;
- Rotas de ligação dos nós, nas quais são especificados o nó de origem, o nó de destino e os links de conexão que unem os dois.

Na Figura 2.3 é apresentado o arquivo application_file. Este arquivo contém *hosts* identificados com os mesmos nomes estabelecidos no arquivo plataform_file e suas funções. Aqueles que possuem a função de *slave* não possuem nenhum atributo, já o *master* possui a informação do número de tarefas que receberá, o poder de cada nó e quanto de comunicação cada um utilizará.

```
<?xml version='1.0'?>
<!DOCTYPE platform_description SYSTEM "surfxml.dtd">
<platform_description version="1">
  <cpu name="gspd-fe" power="10000"/>
  <cpu name="gspd-node1" power="10000"/>
  <cpu name="gspd-node2" power="10000"/>
  <cpu name="gspd-node3" power="10000"/>
  <cpu name="gspd-node4" power="10000"/>
  <cpu name="gspd-node5" power="10000"/>
  <cpu name="gspd-node6" power="10000"/>
  <cpu name="gspd-node7" power="10000"/>
  <cpu name="gspd-node8" power="10000"/>
  <network_link name="lan" bandwidth="100" latency="0.001"/>
  <network_link name="lan1" bandwidth="100" latency="0.001"/>
  <network_link name="lan2" bandwidth="100" latency="0.001"/>
  <network_link name="lan3" bandwidth="100" latency="0.001"/>
  <network_link name="lan4" bandwidth="100" latency="0.001"/>
  <route src="gspd-fe" dst="gspd-node1"><route_element
name="lan1"/></route>
  <route src="gspd-node1" dst="gspd-fe"><route_element
name="lan1"/></route>
  <route src="gspd-fe" dst="gspd-node2"><route_element
name="lan2"/></route>
  <route src="gspd-node2" dst="gspd-fe"><route_element
name="lan2"/></route>
  <route src="gspd-fe" dst="gspd-node3"><route_element
name="lan3"/></route>
  <route src="gspd-node3" dst="gspd-fe"><route_element
name="lan3"/></route>
  <route src="gspd-fe" dst="gspd-node4"><route_element
name="lan4"/></route>
  <route src="gspd-node4" dst="gspd-fe"><route_element
name="lan4"/></route>
</platform_description>
```

Figura 2.2: Arquivo de especificação da plataforma de simulação.

Na Figura 2.4 é apresentado o arquivo que implementa a simulação, ou seja, nele são implementadas as funções próprias do SimGrid que fazem com que a simulação seja executada. Algumas dessas funções são: mestre, escravo e escalonador. Além disso, nesse arquivo é implementado o algoritmo de escalonamento que será utilizado para atribuir as tarefas aos nós.

Nesse arquivo são utilizadas rotinas do MSG como, por exemplo, MSG_create_environment e MSG_launch_application, dentro da função test_all, pois esta constitui o núcleo de simulação, ou seja, esta função recebe como argumentos os arquivos de entrada (plataform_file e application_file) e cria toda a simulação a partir do que está descrito dentro destes arquivos.

```
<?xml version='1.0'?>
<!DOCTYPE platform_description SYSTEM "surfxml.dtd">
<platform_description version="1">
  <process host="gspd-fe" function="master">
     <argument value="20"/>
     <argument value="20000"/>
     <argument value="20"/>
     <argument value="gspd-node1"/>
     <argument value="gspd-node2"/>
     <argument value="gspd-node3"/>
     <argument value="gspd-node4"/>
     <argument value="gspd-node5"/>
     <argument value="gspd-node6"/>
     <argument value="gspd-node7"/>
     <argument value="gspd-node8"/>
  </process>
  <process host="gspd-node1" function="slave"/>
  <process host="gspd-node2" function="slave"/>
  <process host="gspd-node3" function="slave"/>
  <process host="gspd-node4" function="slave"/>
  <process host="gspd-node5" function="slave"/>
  <process host="gspd-node6" function="slave"/>
  <process host="gspd-node7" function="slave"/>
  <process host="gspd-node8" function="slave"/>
</platform_description>
```

Figura 2.3: Arquivo de descrição da aplicação a ser simulada.

Na Figura 2.5 é apresentado o resultado da simulação. O resultado exibe os tempos de ocorrência de cada evento, quem efetuou o evento e uma mensagem referente ao evento. Essas mensagens são inseridas pelo usuário que está efetuando a simulação dentro do algoritmo construído em linguagem C através de rotinas MSG como, por exemplo, INFO0, INFO1 e INFO2.

```
#include <stdio.h>
#include "msg/msg.h"
#include "xbt/sysdep.h"
#include "xbt/log.h"
#include "xbt/asserts.h"
XBT_LOG_NEW_DEFAULT_CATEGORY(msg_test,"Messages specific for this msg
example");
int master(int argc, char *argv[]);
int slave(int argc, char *argv[]);
int forwarder(int argc, char *argv[]);
MSG_error_t test_all(const char *platform_file, const char
*application_file);
```

Figura 2.4: Protótipos do arquivo de simulação do SimGrid.

```
[     0.000][   gspd-fe:master    ] Obtido 8 slaves e 20 tarefas para
processo
[     0.000][   gspd-fe:master    ] Enviando "Task_0" para "gspd-node1"
[     0.201][   gspd-fe:master    ] Enviado
[     0.201][   gspd-fe:master    ] Enviando "Task_1" para "gspd-node2"
[     0.201][gspd-node1:slave     ] Received "Task_0"
[     0.201][gspd-node1:slave     ] Processing "Task_0"
...
[     2.201][gspd-node1:slave     ] "Task_0" done
[     2.402][gspd-node1:slave     ] Received "Task_8"
...
[     5.206][   gspd-fe:master    ] Enviado
[     5.206][   gspd-fe:master    ] Todas as tarefas foram expedidas.
...
[     6.604][gspd-node1:slave     ] Received "finalize"
[     6.604][gspd-node1:slave     ] I'm done. See you!
[     6.804][gspd-node2:slave     ] "Task_17" done
[     6.805][gspd-node2:slave     ] Received "finalize"
[     6.805][gspd-node2:slave     ] I'm done. See you!
...
[     7.211][   gspd-fe:master    ] Goodbye now!
[     7.211][gspd-node8:slave     ] Received "finalize"
[     7.211][gspd-node8:slave     ] I'm done. See you!
[7.211000] msg/global.c:425: [msg_kernel/INFO] Congratulations !
Simulation terminated : all processes are over
[7.211000] cluster.c:182: [msg_test/INFO] Tempo de Simulação 7.211
```

Figura 2.5: Parte do resultado da simulação do SimGrid.

2.2.2 Bricks

O Bricks é um simulador de avaliação de desempenho para programação de algoritmos e de estrutura de sistemas de computação de alto desempenho [Takefusa *et al*., (1999)].

O simulador trabalha com conceito de filas para executar a simulação em um ambiente distribuído. As redes e os servidores são representados por filas e as taxas de serviços dessas filas representam as larguras de banda de cada uma das redes e o poder de processamento de cada servidor.

A arquitetura do Bricks consiste basicamente em um ambiente computacional global e uma unidade de escalonamento, que coordenam o comportamento do sistema [Oliveira, (2008)].

Na Figura 2.6 é apresentada a arquitetura do Bricks.

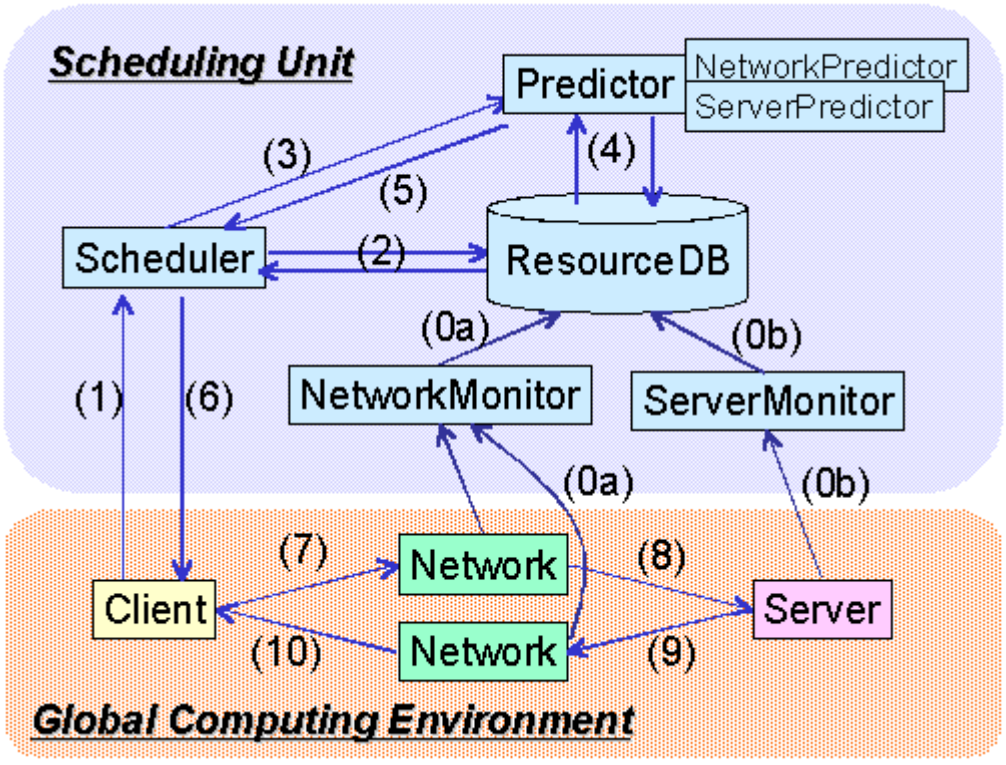

Figura 2.6: Arquitetura do Bricks [Oliveira, (2008)].

O ambiente computacional global representa a plataforma sobre a qual a simulação será realizada.

- *Network*: representa a interconexão de rede do cliente e do servidor,

parametrizados por largura de banda, congestionamento e variação ao longo do tempo;

- *Client*: representa o usuário máquina, na qual as tarefas de computação são iniciadas pelo usuário do programa;
- *Server*: representa os recursos computacionais de um determinado sistema de computação, parametrizado por desempenho, carga e sua variação ao longo do tempo.

A unidade de escalonamento é constituída dos seguintes aspectos:

- *Network Monitor*: mede a largura de banda e latência dos canais de comunicação da grade;
- *Server Monitor*: mede o desempenho, carga de trabalho e a disponibilidade dos servidores;
- *Resource* DB: atua como um banco de dados para informações úteis ao escalonamento e armazena as medições do *Network Monitor* e do *Server Monitor*;
- *Predictor*: recupera as informações a respeito dos recursos em um determinado intervalo de tempo e realiza previsões sobre a disponibilidade de recurso;
- *Scheduler*: aloca uma determinada tarefa invocada pelo usuário ao servidor adequado, tomando decisões com base nas informações sobre recursos e previsões providas pelo *Predictor* e *Resource* DB.

A modelagem da plataforma da grade computacional é feita através da alteração de um arquivo escrito em uma linguagem de *script* própria do Bricks, possibilitando a especificação da topologia da rede, o conjunto de máquinas cliente e servidor, e os parâmetros que caracterizam os recursos computacionais e os canais de comunicação.

2.2.3 OptorSim

O OptorSim é projetado para escalonamentos que utilizam o conceito de réplica de dados, ou seja, trata-se de um simulador voltado para grades de dados, nos

quais as transferências de dados constituem um importante fator limitante do desempenho das tarefas executas.

Com a arquitetura baseada no EU *DataGrid Project* [EU DataGrid Project, (2001)], a simulação é construída assumindo que a grade consiste em vários *sites*, cada um pode prover recursos computacionais e de armazenamento para os trabalhos submetidos. Cada *site* é composto de zero ou mais elementos de computação ou zero ou mais elementos de armazenamento [Bell *et al.*, (2003)].

As decisões sobre movimentação de dados entre *sites* são tomadas pelo *Replica Manager*. Dentro deste componente, a decisão de criar ou eliminar réplicas é controlada pelo *Replica Optimizer*, também chamado Optor. O ponto central do Optor é o algoritmo de otimização de réplicas [Oliveira, (2008)]. Na Figura 2.7 é apresentada a arquitetura do OptorSim.

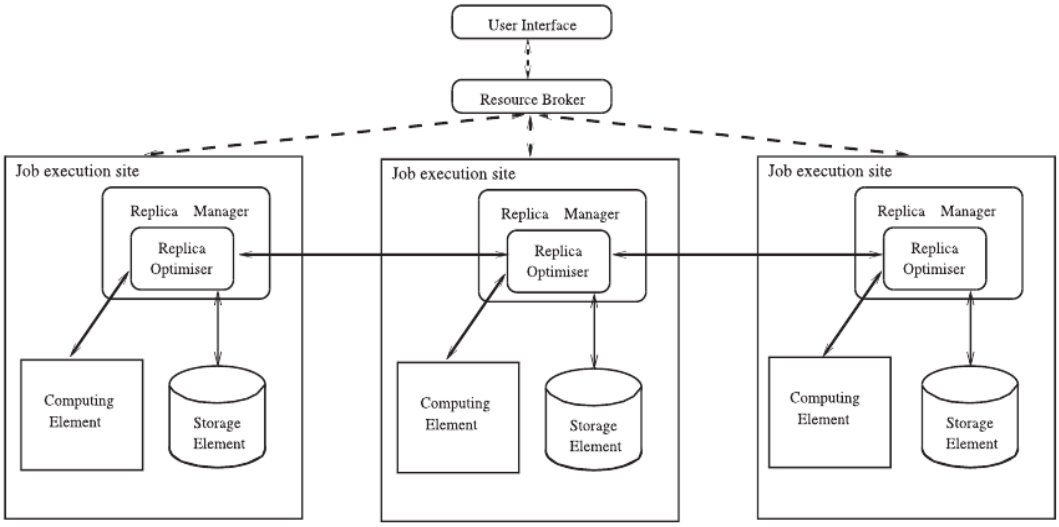

Figura 2.7: Arquitetura do OptorSim [Bell *et al.*, (2003)].

O OptorSim utiliza três arquivos de configuração para efetuar a simulação. A plataforma da grade a ser simulada no Optorsim é especificada em um arquivo de configurações (grid.conf), o qual é lido pela aplicação. Neste arquivo descreve-se a topologia de cada *site* e detalhes de seus componentes dispostos em uma matriz como, por exemplo, o número de elementos computacionais do *site*, o número de armazenamento do *site*, o tamanho dos elementos de armazenamento em *megabytes* e a indicação de tamanho máximo da largura de banda do canal.

As tarefas a serem executadas são indicadas no arquivo jobs.conf, na qual são especificadas informações sobre as tarefas e os arquivos utilizados por elas e a política de cada *site*. O último arquivo contém os parâmetros de entrada como, por exemplo, o caminho para os dois arquivos citados acima, o tipo de usuário, a estratégia de escalonamento que será utilizada, o algoritmo de otimização, a distribuição inicial dos arquivos, o intervalo de tempo entre as submissões de tarefas, entre outras informações.

2.2.4 GridSim

Nesta seção será feita uma breve descrição das principais características e funcionalidades do simulador GridSim.

O GridSim, atualmente na versão 4.1, facilita a simulação de recursos heterogêneos, usuários, escalonadores e aplicações [Oliveira, (2007)]. Suas principais características e funcionalidades são listadas abaixo:

- Permite a modelagem de recursos heterogêneos;
- Os recursos podem ser modelados de forma a serem compartilhados no tempo e no espaço;
- É utilizado o padrão *benchmarks Standard Performance Evaluation Corporation* (SPEC) para definir a capacidade dos recursos;
- O fuso-horário não influencia nos recursos;
- Finais de semana e feriados podem ser mapeados, dependendo do horário local do recurso para aceitar carga de trabalho não pertencente ao *grid*;
- Recursos podem ser pré-reservados;
- *Grids* de dados podem ser simulados;
- O simulador possibilita especificar diferentes topologias de rede para interconectar os recursos e entidades;
- Tarefas podem ser heterogêneas;
- Não há limite de tarefas que podem ser alocadas a um determinado recurso;
- Múltiplos usuários podem enviar tarefas simultaneamente para um mesmo recurso;

- Possibilidade de utilizar escalonadores estáticos ou dinâmicos na simulação;
- O simulador pode gravar estatísticas dos resultados da simulação, as quais podem ser analisadas posteriormente pelos métodos de análise estatística do GridSim.

2.3 Redes de Filas

Uma rede de filas é constituída de centros de serviço e um conjunto de usuários, os quais recebem serviços nos centros. Um centro de serviço é constituído de um ou mais servidores que prestam serviços aos usuários, e uma fila de usuários que aguardam serem atendidos quando a capacidade do centro de serviço é esgotada [Soares, (1992)].

Os centros de serviço podem ser classificados pelo número de servidores e o número de filas de espera para um dado servidor. Assim, pode-se ter centros com uma fila (ou várias) e um servidor (ou vários).

O modelo mais simples é constituído de um centro de serviço com uma fila e um servidor. Os clientes chegam a algum servidor e se ele estiver ocupado, o cliente aguarda no final da fila associada com aquele servidor. Este cliente será selecionado para atendimento de acordo com a disciplina da fila. O serviço requisitado é então executado pelo servidor e ao seu término o cliente deixa o centro de serviço.

Na Figura 2.8 é apresentado um centro de serviço constituído de 1 fila e 1 servidor.

Há outras variações do modelo básico como, por exemplo, um centro de serviço constituído de uma fila que alimenta vários servidores, ou múltiplas filas associadas a um único servidor, ou ainda múltiplas filas e múltiplos servidores.

Para cada um dos modelos descritos, pode-se identificar alguns parâmetros necessários para a especificação completa do modelo. No modelo básico (uma fila e um servidor) deve-se definir qual a disciplina da fila, a taxa de chegada dos usuários e o tempo necessário para executar o serviço. O modelo constituído de uma fila e vários servidores necessita, além dos parâmetros do modelo mais simples, da seleção de um servidor quando mais de um está desocupado. Para o modelo constituído de

múltiplas filas e um servidor, além dos parâmetros necessários ao primeiro, é necessário especificar a maneira pela qual uma fila é selecionada quando o servidor fica disponível e como um cliente escolhe uma fila para aguardar pelo serviço. O último modelo (múltiplas filas e múltiplos servidores) necessita de todos os parâmetros identificados para os outros modelos.

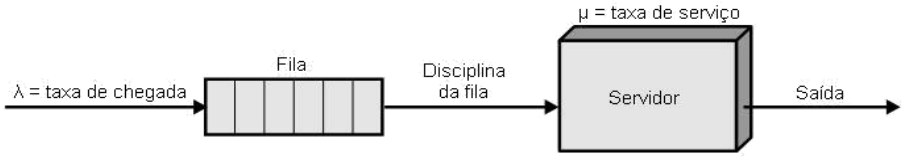

Figura 2.8: Centro de serviço – Uma fila e um servidor.

2.4 Projeto de Interface

Nesta seção é apresentado o estudo sobre projetos de interface. Isso envolve o estudo da interação humano-computador, estudo das características que um projeto de interface deve possuir e o estudo de uma linguagem de modelagem que será usada para a descrição das ações da interface.

2.4.1 Interação Humano-Computador

Para o desenvolvimento de um sistema computacional, a interface com o usuário é um requisito de importância vital e o estudo da interação humano-computador auxilia no desenvolvimento da interface, de modo que os usuários sintam-se satisfeitos ao utilizar do sistema.

O estudo da interação humano-computador considera quatro elementos básicos: o sistema, os usuários, os desenvolvedores e o ambiente de uso [Dix *et al.*, (2004)].

A interação humano-computador provê bases teóricas para a elaboração de uma interface que seja segura, eficiente e que traga satisfação ao usuário. A interação humano-computador pode se feita em diferentes estilos e cada um desses estilos tem suas características próprias. Alguns exemplos de estilos incluem: linhas de

comando, menus, linguagem natural, interfaces de questionamento, formulários, planilhas eletrônicas, WIMP (*Windows, Icons, Menus and Pointers*), tri-dimensional, entre outros [Dix *et al*., (2004)].

A usabilidade e a comunicabilidade são conceitos que fazem parte do estudo da interação humano-computador. A comunicabilidade está relacionada com a eficiência da interação, ou seja, quando o usuário efetua uma ação no sistema esperando uma resposta, a resposta mostrada ao usuário tem que ser clara e objetiva, principalmente quando forem exibidas mensagens de erros, pois é necessário que o usuário saiba por que aquela ação não pode ser realizada. A usabilidade é a facilidade e a simplicidade com que a interface pode ser usada, levando em conta aspectos como, por exemplo, facilidade de aprendizado, facilidade de uso, satisfação do usuário, flexibilidade e produtividade.

2.4.2 Engenharia Cognitiva

A Engenharia Cognitiva estuda a cognição, processo pelo qual se adquire conhecimento e aplica suas teorias na compreensão das capacidades e limitações da mente dos usuários. Com esses conhecimentos o desenvolvedor cria seu modelo mental (“modelo de *design*”), que será implementado visando à interação do usuário com ele. Nessa interação o usuário cria seu modelo mental do que espera que a aplicação faça (modelo do usuário), efetuando a partir deste modelo suas ações e objetivos [De Souza *et al*., (1999)].

Para se criar um “modelo de usuário” consistente com o “modelo de *design*”, o desenvolvedor precisa entender o processo pelo qual o usuário interage com a interface antes de criar seu modelo. E, para entender o processo de interação do usuário com o sistema, utiliza-se a teoria da ação.

Na Figura 2.9 são apresentados os dois golfos que identificam cada passo seguido pelo usuário na interação com o sistema.

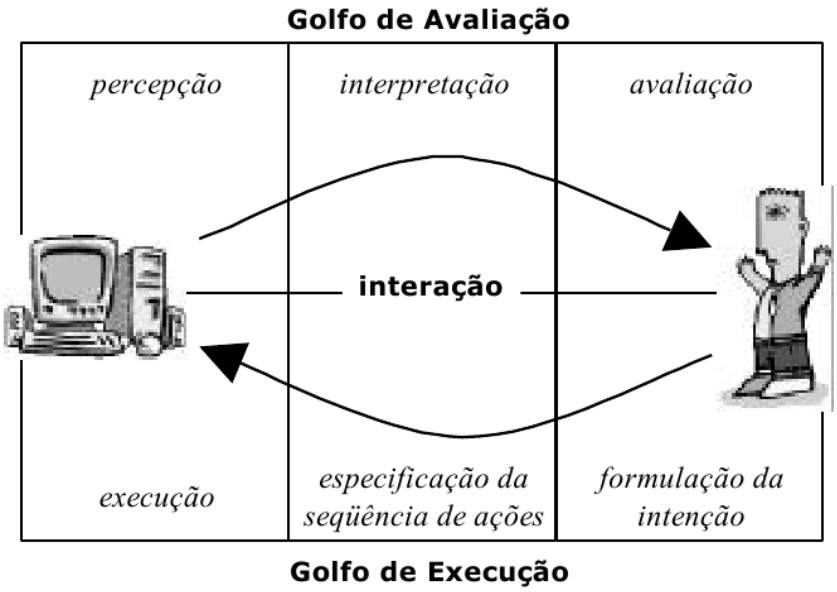

Figura 2.9: Diagrama da teoria da ação [De Souza *et al.*, (1999)].

Os golfos definidos na teoria da ação são os seguintes [De Souza *et al.*, (1999)]:

- Golfo de Execução: neste golfo o usuário formula as metas que deseja alcançar (formulação da intenção). A fim de atingir essas metas, ele começa a definir as ações que serão executadas (especificação da seqüência de ações). Até este ponto, o usuário faz isso mentalmente. Na terceira etapa, já com as ações definidas, ele começa a execução destas ações na aplicação (execução). Note que essas ações não precisam necessariamente ocorrer em sua totalidade para que as próximas sejam realizadas, normalmente os usuários tendem a executar ações na interface enquanto ainda buscam definir as ações que desejam executar;
- Golfo de Avaliação: após uma ação do usuário, a interface muda de estado. Nesse momento o usuário tem a percepção da mudança de estado da interface, começa a interpretar o que aconteceu no sistema e avaliar se o resultado da alteração foi o que ele esperava que acontecesse. É importante ressaltar que se não houver nenhuma modificação da interface na etapa de percepção do usuário, ele passa a achar que a ação não produziu resultados, assumindo que sua meta não foi atingida.

2.4.3 Características Para um Projeto de Interface

Algumas características devem ser consideradas no projeto de interface [Sharp *et al.*, (2007)]:

- Consistência: a interface deve apresentar um comportamento previsível;
- Facilidade de aprendizagem e utilização: a interface deve proporcionar facilidade de aprendizagem para os usuários iniciantes e ser eficiente para os mais experientes;
- Ação de respostas: para toda ação do usuário, a interface deve produzir uma resposta, ou seja, deve mudar seu estado;
- Satisfação: esta característica está relacionada com atingir o resultado esperado pelo usuário;
- Manifestação de erros: a interface deve apresentar para o usuário notificações de execuções que apresentem erros;
- Suporte: a interface deve proporcionar ao usuário garantias de que o que ele inseriu no sistema seja mantido caso ocorra algum erro ou caso o sistema fique ocioso por muito tempo;
- Familiaridade e Simplicidade: a interface deve proporcionar ao usuário todos os elementos necessários para o seu uso, de modo que não sobrecarregue sua visualização;
- Incentivo: cada ação do usuário deve promover uma reação de acordo com o que o usuário deseja;
- Acessibilidade: todos os objetos da interface têm que estar à disposição do usuário;
- Versatilidade: essa característica permite que o usuário determine o método de interação com a interface.

2.5 Considerações Finais

Neste capítulo foram abordados os principais simuladores de grades computacionais existentes a fim de identificar os principais parâmetros necessários

para a construção de um simulador de grades computacionais. Em seguida, foi feita uma breve discussão a respeito de redes de filas. Além disso, discutiram-se alguns princípios, técnicas e conceitos para o projeto da interface icônica do simulador.

Capítulo 3 – Atividades Realizadas

3.1 Considerações Iniciais

Este capítulo trata da especificação das gramáticas para o modelo utilizado pelo motor de simulação, das gramáticas para o modelo icônico, das gramáticas de modelos SimGrid, da especificação e construção do protótipo do interpretador de modelos externos e da especificação e implementação da interface icônica.

A especificação das gramáticas foi feita através da notação *Backus-Naur Form* (BNF).

A implementação de todo o simulador foi feita utilizando-se a linguagem Java com o compilador JDK 1.6.0 build 21 e para as implementações dos analisadores léxico, sintático e semântico e do interpretador de modelos SimGrid utilizou-se também a ferramenta JavaCC 5.0.

3.2 Especificação das Gramáticas Para Modelos Simuláveis

Nesta seção serão apresentadas as gramáticas regular e livre de contexto para o modelo simulável. Palavras reservadas estão representadas em negrito.

3.2.1 Gramática Regular

Foram definidos dígitos, letras, símbolos (“.”, “,” e “;”), números inteiros, números reais e caracteres especiais (<especiais>). Além disso, foi estabelecido que comentários terão a mesma sintaxe que a linguagem C (“/* {qualquer caracter} */” ou “// {qualquer caracter}”).

A gramática regular é apresentada a seguir.

<comentários> ::= /* {qualquer caracter} */ | // {qualquer caracter}

<letra> ::= a | b | c | ... | z | A | B | C |... | Z

<dígito> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<vírgula> ::= “,”

<ponto_vírgula> ::= “;”

<ponto> ::= “.”

<especiais> ::= “!” | “@” | “#” | “$” | “%” | “&” | “(“ | “)” | “-“ | “_” | “+” | “=” | “{“ | “}” | “[“ | “]” | “~” | “^” | “’” | “ç” | “ª” | “º” | “.” | “;” | “,” | “:” | “|” | “/” | “\”

3.2.2 Gramática Livre de Contexto

Os parâmetros de simulação (<definições_simulação>) envolvem os modelos de tarefa (<modelo_tarefa>) que poderão ser utilizados. Estes modelos incluem: modelo de carga (<modelo_carga>), modelo de *trace* (<modelo_trace) ou modelo aleatório (<random>).

O modelo de carga é representado pela palavra reservada **MAQUINA** e recebe como parâmetros o número de tarefas (<num_tarefas>), o tamanho mínimo (<min_comp>) e máximo (max_comp) de cada tarefa e o tamanho mínimo (<min_comm>) e máximo (<max_comm>) de comunicação que cada servidor utilizará.

O modelo de *trace* é representado pela palavra reservada **TRACE** e por um nome de arquivo (<arquivo_ID>), que contém eventos que deverão ser executados pelo simulador.

O modelo *random* é representado pela palavra reservada **RANDOM** e escolhe aleatoriamente os parâmetros para as tarefas. Sendo que esses parâmetros envolvem a capacidade de processamento mínima (<cap_proc_min>), média (<cap_proc_med>) e máxima (<cap_proc_max>), taxa de probabilidade de processamento (<prob_proc>), capacidade de comunicação mínima (<cap_comm_min>), média (<cap_comm_med>) e máxima (<cap_comm_max>), taxa de probabilidade de comunicação (<prob_comm>) e tempo de chegada mínimo (<t_chegada_min>), médio (<t_chegada_med>) e máximo (<t_chegada_max>).

Os centros de serviços que representam as redes de filas foram divididos em quatro categorias. Cada uma delas representa um ícone da interface icônica.

Cada fila, servidor e centro de serviço possui um identificador único. Sendo eles, respectivamente: <fila_ID>, <servidor_ID> e <CS_ID>. Estes identificadores podem ser representados por letras e dígitos, mas sempre começando por uma letra.

Para definir uma fila é necessário apenas definir o identificador para a fila após a palavra chave **FILAS** dentro da declaração um centro de serviço, sendo o número de identificadores de filas compatível com o número de filas para o centro de serviço.

Cada centro de serviço recebe seus respectivos parâmetros.

O <CS_0> representa o ícone do tipo máquina, e cada servidor declarado precisa receber um do identificador, a capacidade de processamento (<capacidade_processamento>), a taxa de ocupação (<taxa_ocupacao>) e uma função (mestre ou escravo). Se o servidor for escravo, necessita também de um algoritmo de escalonamento (<disciplina>) e a lista de escravos (<nos>).

O <CS_1> representa o ícone do tipo *cluster*. Além das filas e servidores, recebe também um algoritmo de escalonamento. Os servidores para este centro de

serviço (<CS_1>) recebem um valor de poder computacional (<capacidade_processamento>), banda (<banda>) e latência (<latencia>).

O <CS_2> representa o ícone do tipo rede, seus servidores recebem os parâmetros banda, taxa de ocupação e latência.

O <CS_3> representa o ícone do tipo *internet* e seus servidores recebem os mesmos parâmetros do centro de serviço <CS_2>.

A gramática livre de contexto é apresentada a seguir.

<inteiro> ::= {<dígito>}+

<real> ::= {<inteiro>}+ <ponto> {<inteiro>}+

<identificador> ::= <letra> {<letra> | <dígito> | <especial>}*

<min_comp> ::= <real>

<max_comp> ::= <real>

<min_comm> ::= <real>

<max_comm> ::= <real>

<cap_proc_min> ::= <inteiro>

<cap_proc_med> ::= <inteiro>

<cap_proc_max> ::= <inteiro>

<prob_proc> ::= <real>

<cap_comm_min> ::= <inteiro>

<cap_comm_med> ::= <inteiro>

<cap_comm_max> ::= <inteiro>

<prob_comm> ::= <real>

<t_chegada_min> ::= <inteiro>

<t_chegada_med> ::= <inteiro>

<t_chegada_max> ::= <inteiro>

<definições_simulação> ::= TAREFA <modelo_tarefa> FIM_TAREFA

<modelo_tarefa> ::= <modelo_carga> | <modelo_trace> | <random>

<modelo_carga> ::= **MAQUINA** (<servidor_ID> <num_tarefas> <min_comp> <max_comp> <min_comm> <max_comm>)+

<modelo_trace> ::= **TRACE** <arquivo_ID>

<random> ::= **RANDOM** <cap_proc_min> <cap_proc_med> <cap_proc_max> <prob_proc> <cap_comm_min> <cap_comm_med> <cap_comm_max> <prob_comm> <t_chegada_min> <t_chegada_med> <t_chegada_max>

<arquivo_ID> ::= <identificador>

<fila_ID> ::= <identificador>

<servidor_ID> ::= <identificador>

<CS_ID> ::= <identificador>

<num_servidores> ::= <inteiro>

<num_filas> ::= <inteiro>

<tipo_servidor> ::= <inteiro>

<capacidade_processamento> ::= <real>

<taxa_ocupacao> ::= <real>

<banda> ::= <real>

<latencia> ::= <real>

<disciplina> ::= RR

<filas> ::= <filas> <fila_ID> | <fila_ID>

<nos> ::= <servidor_ID> <nos> | <servidor_ID>

<mestre> ::= **MESTRE** <disciplina> **LMAQ** <nos> | **ESCRAVO**

<servidor1> ::= <servidor_ID> <tipo_servidor> <capacidade_processamento> <taxa_ocupacao> <mestre>

<servidor2> ::= <servidor_ID> <tipo_servidor> <capacidade_processamento> <banda> <latencia>

<servidor3> ::= <servidor_ID> <tipo_servidor> <banda> <taxa_ocupacao> <latencia>

<parâmetros> ::= <num_filas> <num_servidores>

<CS_0> ::= CS_0 <CS_ID> <parâmetros> FILAS <filas> SERVIDORES <servidor1>

<CS_1> ::= CS_1 <CS_ID> <parâmetros> <disciplina> FILAS <filas> SERVIDORES <servidor2>

<CS_2> ::= CS_2 <CS_ID> <parâmetros> FILAS <filas> SERVIDORES <servidor3>

<CS_3> ::= CS_3 <CS_ID> <parâmetros> FILAS <filas> SERVIDORES <servidor3>

<CS> ::= <CS_0> | <CS_1> | <CS_2> | <CS_3>

<lista_CS> ::= <lista_CS> <CS> | <CS>

<defina_CS> ::= CENTROS_DE_SERVICOS <lista_CS> FIM_CENTROS_DE_SERVICOS

<lista_conexoes> ::= <CS_ID> <CS_ID> | <lista_conexoes> <CS_ID> <CS_ID>

<conexoes> ::= CONEXOES <lista_conexoes> FIM_ CONEXOES

<definicoes> ::= <defina_CS> | <conexoes> | <definições_simulação>

<lista_definicoes> ::= <lista_definicoes> <definicoes> | <definicoes>

<modelo_filas> ::= MODELO <lista_definicoes> FIM_MODELO

Exemplo de um modelo simulável

Na Figura 3.1 é apresentado um exemplo de modelo simulável que foi construído a partir das gramáticas regular e livre de contexto especificadas para a utilização no motor de simulação.

```
MODELO
TAREFA
RANDOM 10 100 1000 1.0
10 100 1000 1.0
10 100 1000
FIM_TAREFA
CENTROS_DE_SERVICOS
CS_2 cs_lan9 1 1 FILAS fila_lan9 SERVIDORES serv_lan9 1 1000.0 0.9 0.02
CS_0 cs_icon7 1 1 FILAS fila_icon7 SERVIDORES serv_icon7 0 1000.0 0.2 ESCRAVO
CS_0 cs_icon0 1 1 FILAS fila_icon0 SERVIDORES serv_icon0 0 1000.0 0.1 MESTRE
RR LMAQ icon1 icon7
CS_2 cs_lan4 1 1 FILAS fila_lan4 SERVIDORES serv_lan4 1 1000.0 0.9 0.02
CS_2 cs_icon10 1 1 FILAS fila_icon10 SERVIDORES serv_icon10 1 1000.0 0.9 0.02
CS_1 cs_icon1 2 100 RR FILAS fila_0_icon1 fila_1_icon1 SERVIDORES serv_icon1 0
1000.0 100.0 0.2
CS_3 cs_icon2 1 1 FILAS fila_icon2 SERVIDORES serv_icon2 1 1000.0 0.2 0.1
FIM_CENTROS_DE_SERVICOS
CONEXOES
cs_icon2        cs_lan9
cs_lan9         cs_icon7
cs_icon2        cs_lan4
cs_lan4         cs_icon1
cs_icon0        cs_icon10
cs_icon10       cs_icon2
FIM_CONEXOES
FIM_MODELO
```

Figura 3.1: Exemplo de modelo simulável.

3.3 Especificação das Gramáticas Para Modelos Icônicos

Nesta seção serão apresentadas as gramáticas regular e livre de contexto para o modelo utilizado na interface icônica.

3.3.1 Gramática Regular

Na gramática regular foram definidos dígitos, letras e caracteres especiais. Além disso, foi estabelecido que comentários serão representados por "#" (<comentários>).

A gramática regular é apresentada a seguir.

<comentários> ::= “#”{qualquer caracter}*

<letra> ::= a | b | c | ... | z | A | B | C |... | Z

<dígito> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<especiais> ::= “!” | “@” | “#” | “$” | “%” | “&” | “(“ | “)” | “-“ | “_” | “+” | “=” | “{“ | “}” | “[“ | “]” | “~” | “^” | “’” | “ç” | “ª” | “º” | “;” | “.” | “,”

3.3.2 Gramática Livre de Contexto

Na gramática livre de contexto para o modelo icônico foram definidos números inteiros e reais, identificadores, tipos de ícones e os parâmetros de cada um. Além disso, palavras reservadas estão representadas em negrito.

Para o ícone “servidor” (<no>) define-se o seu identificador, poder computacional, taxa de ocupação e sua função (mestre ou escravo). Se o servidor for um mestre, definem-se outros parâmetros, tais como, algoritmo de escalonamento e seus escravos. E se o servidor for escravo, não há parâmetros extras.

O ícone “*cluster*” possui identificador, número de escravos, poder computacional, latência, taxa de ocupação e algoritmo de escalonamento.

Já o ícone “*link*” possui identificador, banda, latência, taxa de ocupação e as duas máquinas conectadas no *link* (ponto a ponto).

O ícone “*internet*” (<inet>) possui os mesmos parâmetros do ícone *link*, a única diferença é que, ao invés de conectar máquinas, ele conecta redes.

Por último, o ícone “carga de trabalho” (<carga>), possui o tipo de carga de trabalho, que pode ser do tipo máquina, *random* ou *trace*, e seus parâmetros. A carga do tipo máquina possui o identificador do mestre que recebe a carga de trabalho, o número de tarefas, o tamanho máximo e mínimo da carga da tarefa e da comunicação e a forma de distribuição da carga de trabalho. Já a carga *random* gera uma carga aleatória e seus parâmetros são: capacidade de processamento mínima, média e máxima, taxa de probabilidade de processamento, capacidade de comunicação mínima, média e máxima, taxa de probabilidade de comunicação e tempo de chegada

mínimo, médio e máximo. Por último, a carga *trace* possui como único parâmetro o arquivo a ser lido.

<inteiro> ::= {<dígito>}$^{+}$

<real> ::= {<inteiro>}<ponto>{<inteiro>}

<identificador> ::= <letra> {<letra> | <dígito> | <especiais>}*

<servidor_ID> ::= <identificador>

<link_ID> ::= <identificador>

<cluster_ID> ::= <identificador>

<inet_ID> ::= <identificador>

<arquivo_ID> ::= <identificador>

<num_escravos> ::= <inteiro>

<num_tarefas> ::= <inteiro>

<max_comp_tam_tarefa> ::= <real>

<min_comp_tam_tarefa> ::= <real>

<max_comm_tam_tarefa> ::= <real>

<min_comm_tam_tarefa> ::= <real>

<poder_computacional> ::= <real>

<taxa_ocupação> ::= <real>

<banda> ::= <real>

<latência> ::= <real>

<origem> ::= <identificador>

<destino> ::= <identificador>

<cap_proc_min> ::= <inteiro>

<cap_proc_med> ::= <inteiro>

<cap_proc_max> ::= <inteiro>

<prob_proc> ::= <real>

<cap_comm_min> ::= <inteiro>

<cap_comm_med> ::= <inteiro>

<cap_comm_max> ::= <inteiro>

<prob_comm> ::= <real>

<t_chegada_min> ::= <inteiro>

<t_chegada_med> ::= <inteiro>

<t_chegada_max> ::= <inteiro>

<modelo> ::= <ícones>

<ícones> ::= {<ícone>}+

<ícone> ::= <no> | <cluster> | <link> | <inet> | <carga>

<notipo> ::= MESTRE <clusteralg>LMAQ<escravos> | ESCRAVO

<escravos> ::= {<servidor_ID>}+

<no> ::= MAQ<servidor_ID><poder_computacional><taxa_ocupação><notipo>

<link>::= REDE<link_ID><banda><latência><taxa_ocupação>CONECTA<origem><destino>

<clusteralg> ::= RR | WORKQUEUE | FPLTF

<cluster> ::= CLUSTER <cluster_ID> <num_escravos> <poder_computacional> <latência> <taxa_ocupação> <clusteralg>

<inet> ::= INET<inet_ID><banda><latência><taxa_ocupação>

<carga> ::= CARGA <tipo_carga>

<tipo_carga> ::= <máquina> | <random> | <trace>

<máquina> ::= MAQUINA <servidor_ID> <num_tarefas> <max_comp_tam_tarefa> <min_comp_tam_tarefa> <max_comm_tam_tarefa> <min_comm_tam_tarefa><distribuição>

<random> ::= RANDOM <cap_proc_min> <cap_proc_med> <cap_proc_max> <prob_proc> <cap_comm_min> <cap_comm_med> <cap_comm_max> <prob_comm> <t_chegada_min> <t_chegada_med> <t_chegada_max>

<trace> ::= TRACE<arquivo_ID>

Exemplo de um Modelo Icônico

Na Figura 3.2 é apresentado o modelo icônico correspondente ao modelo simulável da Figura 3.1.

MAQ icon7 1000.0 0.2 **ESCRAVO**
MAQ icon0 1000.0 0.1 **MESTRE RR LMAQ** icon1 icon7
CLUSTER icon1 100 1000.0 100.0 0.2 RoundRobin
INET icon2 1000.0 0.1 0.2
REDE lan9 1000.0 0.02 0.9 **CONECTA** icon2 icon7
REDE lan4 1000.0 0.02 0.9 **CONECTA** icon2 icon1
REDE icon10 1000.0 0.02 0.9 **CONECTA** icon0 icon2
CARGA RANDOM
10 100 1000 1.0
10 100 1000 1.0
10 100 1000

Figura 3.2: Exemplo de modelo icônico.

3.4 Especificação das Gramáticas Para Modelos SimGrid

Nesta seção serão apresentadas as gramáticas regular e livre de contexto para um modelo de simulação do SimGrid.

3.4.1 Gramática Regular

Para especificar a gramática regular do SimGrid, foram identificados dígitos, letras, caracteres especiais e alguns caracteres específicos que são utilizados na sintaxe do modelo.

A gramática regular para um modelo do SimGrid é apresentada a seguir.

<letra> ::= a | b | c | ... | z | A | B | C |... | Z

<dígito> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<ponto> ::= “.”

<barra> ::= “/”

<exclamação> ::= “!”

<interrogação> ::= “?”

<menor> ::= “<”

<maior> ::= “>”

<igual> ::= “=”

<apostrofo> ::= “’”

<aspas> ::= “””

<especiais> ::= “!” | “@” | “#” | “$” | “%” | “&” | “(“ | “)” | “-“ | “_” | “+” | “{“ | “}” | “[“ | “]” | “~” | “^” | “,” | “;” | “ç” | “ª” | “º” | “’” | “=” | “.”

3.4.2 Gramática Livre de Contexto

A gramática livre de contexto do SimGrid inclui comentários, números inteiros e reais e especifica alguns parâmetros necessários para simulação.

Um modelo especificado na plataforma do SimGrid é dividido em 2 arquivos: application_file e plataform_file.

Um deles (<plataform_file>) contém informações sobre a plataforma de simulação, tais como, servidor, rede e roteamento.

O servidor possui um identificador (<servidor_ID) e poder computacional (<capacidade_processamento>), que pode ser dado como número inteiro ou real.

A rede também possui um identificador (<rede_ID>), mas também possui parâmetros de rede, como, por exemplo, largura de banda (<largura_banda>) e latência (<latencia>).

E por último, o roteamento indica o servidor de origem e o servidor de destino para um dado *link*.

O outro arquivo (<application_file>) descreve os processos mestres, escravos e outros tipos de processos.

O processo mestre possui um identificador, número de tarefas, tamanho máximo do tamanho de cada tarefa e um valor máximo de comunicação que cada um utilizará. Opcionalmente, pode-se definir um valor mínimo para o tamanho de cada tarefa e para comunicação.

O processo escravo recebe apenas o necessário para a simulação, um identificador.

Existem ainda outros tipos de processos (<outros>) que não foram utilizados na simulação por se tratar de uma simulação mais complexa e que ainda não possui parâmetros correspondentes no protótipo do simulador proposto e portanto, que não constam ainda no interpretador de modelos externos. No entanto, estes processos já foram listados e descritos na gramática.

A gramática livre de contexto para o modelo do SimGrid é apresentada a seguir.

<comentários> ::= <menor><exclamação> {qualquer caracter}* <maior>

<inteiro> ::= {<dígito>}+

<real> ::= {<inteiro>}<ponto>{<inteiro>}

<capacidade_processamento> ::= <real> | <inteiro>

<largura_banda> ::= <real> | <inteiro>

<latencia> ::= <real> | <inteiro>

<identificador> ::= <letra> {<letra> | <dígito> | <especiais>}*

<servidor_ID> ::= <identificador>

<rede_ID> ::= <identificador>

<modelo> ::= <xml_version> <início_plataforma> (<application_file> | <plataform_file>) <fim_plataforma>

<application_file> ::= {<processos>}+

<plataform_file> ::= {<plataforma>}+

<xml_version> ::= <menor> <interrogação> **xml version** <igual> <apostrofo> <real> <apostrofo> <interrogação> <maior>

<início_plataforma> ::= <menor> **platform_description version** <igual> <aspas> <inteiro> <aspas> <maior>

<fim_plataforma> ::= <menor> <barra> **platform_description** <maior>

<processos> ::= <master> | <slave> | <outros>

<plataforma> ::= <servidor> | <rede> | <roteamento>

<master> ::= <menor> **process host** <igual> <aspas> <servidor_ID> <aspas> **function** <igual> <aspas> **master** <aspas> <maior> <menor> **argument value** <igual> <aspas> <num_tarefas> <aspas> <barra> <maior> ((<menor> **argument value** <igual> <aspas> <max_comp_tam_tarefa> <aspas> <barra> <maior> <menor> **argument value** <igual> <aspas> <min_comp_tam_tarefa> <aspas> <barra> <maior> <menor> **argument value** <igual> <aspas> <max_comm_tam_tarefa> <aspas> <barra> <maior> <menor> **argument value** <igual> <aspas> <min_comm_tam_tarefa> <aspas> <barra> <maior>) | (<menor> **argument value** <igual> <aspas> <max_comp_tam_tarefa> <aspas> <barra> <maior> <menor> **argument value** <igual> <aspas> <max_comm_tam_tarefa> <aspas> <barra> <maior>)) {<menor> **argument value** <igual> <aspas> <servidor_ID> <aspas> <barra> <maior>}$^{+}$ <menor> <barra> <maior>

<slave> ::= <menor> **process host** <igual> <aspas> <servidor_ID> <aspas> **function** <igual> <aspas> **slave** <aspas> <barra> <maior>

<outros> ::= <tasksource> | <slavecomm> | <reloadhost> | <forwarderscheduler> | <forwardernode> | <forwardercomm>

<servidor> ::= <menor> **cpu name** <igual> <aspas> <servidor_ID> <aspas> **power**<igual><aspas> <capacidade_processamento><aspas> <barra><maior>

<rede> ::= <menor> **network_link name** <igual> <aspas> <rede_ID> <aspa> **bandwidth** <igual> <aspas> <largura_banda> <aspas> **latency** <latencia> <aspas> (<real> | <inteiro>) <aspas> <barra> <maior>

<roteamento> ::= <menor> **route src** <igual> <aspas> <servidor_ID> <aspas> **dst** <igual> <aspas> <servidor_ID> <aspas> <maior> <menor> **route_element name** <igual> <aspas> <rede_ID> <aspas> <barra> <maior> <menor> <barra>**route**<maior>

<tasksource> ::= <menor> **process host** <igual> <aspas> <servidor_ID> <aspas> **function** <igual> <aspas> **tasksource** <aspas> <maior> <menor> **argument value** <igual> <aspas> <inteiro> <aspas> <barra> <maior> <menor> <barra> **process** <maior>

<slavecomm> ::= <menor> **process host** <igual> <aspas> <servidor_ID> <aspas> **function** <igual> <aspas> **slavecomm** <aspas> <barra> <maior>

<reloadhost> ::= <menor> **process host** <igual> <aspas> <servidor_ID> <aspas> **function** <igual> <aspas> **reloadhost** <aspas> <barra> <maior>

<forwarderscheduler> ::= <menor> **process host** <igual> <aspas> <servidor_ID> <aspas> **function** <igual> <aspas> **forwarderscheduler** <aspas> <maior> <menor> **argument value** <igual> <aspas> <servidor_ID> <aspas> <barra> <maior> <menor> <barra> **process** <maior>

<forwardernode> ::= <menor> **process host** <igual> <aspas> <servidor_ID> <aspas> **function** <igual> <aspas> **forwardernode** <aspas> <maior> {<menor> **argument value** <igual> <aspas> <servidor_ID> <aspas> <barra> <maior>}$^{+}$ <menor> <barra> **process** <maior>

<forwardercomm> ::= <menor> **process host** <igual> <aspas> <servidor_ID> <aspas> **function** <igual> <aspas> **forwardercomm** <aspas> <maior> {<menor> **argument value** <igual> <aspas> <servidor_ID> <aspas> <barra> <maior>}$^{+}$ <menor> <barra> **process** <maior>

3.5 Prototipação dos Interpretadores

Para a construção dos protótipos do interpretadores foi necessária a construção dos analisadores léxico e sintático para as gramáticas regular e livre de contexto que especificam as linguagens definidas. Para tal, foi utilizada a ferramenta JavaCC 5.0 (Java Compiler Compiler).

O JavaCC é uma ferramenta geradora de analisadores gramaticais que analisa sintaticamente e semanticamente uma linguagem e é capaz de transformá-la em outra.

Suas principais características são [Delamaro, (2004)]:

- Análise *top-down*;
- É extremamente modelável;
- Suporta entrada total em Unicode;
- A especificação léxica e sintática pode ser feita em um único arquivo;
- Possui estados léxicos e ações léxicas;
- Permite análise léxica *case-insensitive*;
- Possui *lookahead* sintático e semântico;
- Possui um pré-processador para construção de árvores de análise, o JJTree [JJTree, (2010)];
- Possui um gerador de documentação, o JJDoc [JJDoc, (2010)];
- Permite a utilização da notação BNF estendida para especificações gramaticais e léxicas;
- Possui extensas operações de depuração;
- Gera relatórios de erros;
- É totalmente compatível e gerador de código Java.

O principal motivo da utilização do JavaCC, dentre as várias ferramentas disponíveis para a construção do protótipo do interpretador de modelos externos, foi a total compatibilidade e geração de código Java, visto que o trabalho desenvolvido neste projeto faz parte do contexto de um projeto de simulador de grades computacionais, que é totalmente desenvolvido em Java.

O interpretador de modelos externos encontra-se especificado no arquivo fonte de extensão .jj, que é submetido a um compilador JavaCC. Em seguida, o

código Java gerado pelo JavaCC é compilado com um compilador Java, e executado pela Java Virtual Machine (JVM). Para este projeto, foi utilizado o compilador JDK 1.6.0 Update 20.

3.5.1 Prototipação do Interpretador de Modelos Simuláveis

Para realizar a interpretação do modelo simulável a fim de gerar os dados para efetuar a simualação, o interpretador realiza uma leitura do arquivo de especificação do modelo simulável, e vai adicionando os centros de serviços diretamente em um objeto da classe RedeDeFilas, instanciado no início do reconhecimento.

Os atributos das tarefas reconhecidas pelo interpretador são concatenados em uma *string* utilizando, para isso, separadores específicos instanciados com os nomes de char253, char252 e char254.

Os escravos de um servidor mestre são armazenados em uma lista de objetos da classe privada Escravos, e serão acessados posteriormente para serem incluídos no objeto da classe RedesDeFilas.

Após todo o arquivo ser reconhecido, todos os nomes de ícones são convertidos em códigos numéricos ordenados. Estes códigos são entendidos pela rede de filas como sendo os identificadores de cada centro de serviço, fila ou servidor.

Se não houver erros nos processos de reconhecimento e validação do modelo simulável, uma resposta é retornado para o usuário indicando que o modelo foi reconhecido com sucesso. Por fim, é iniciada a fase de simulação, onde são passados os objetos da classe RedeDeFilas.

Na Figura 3.3 é apresentado o diagrama de classes UML do protótipo do interpretador de modelos simuláveis.

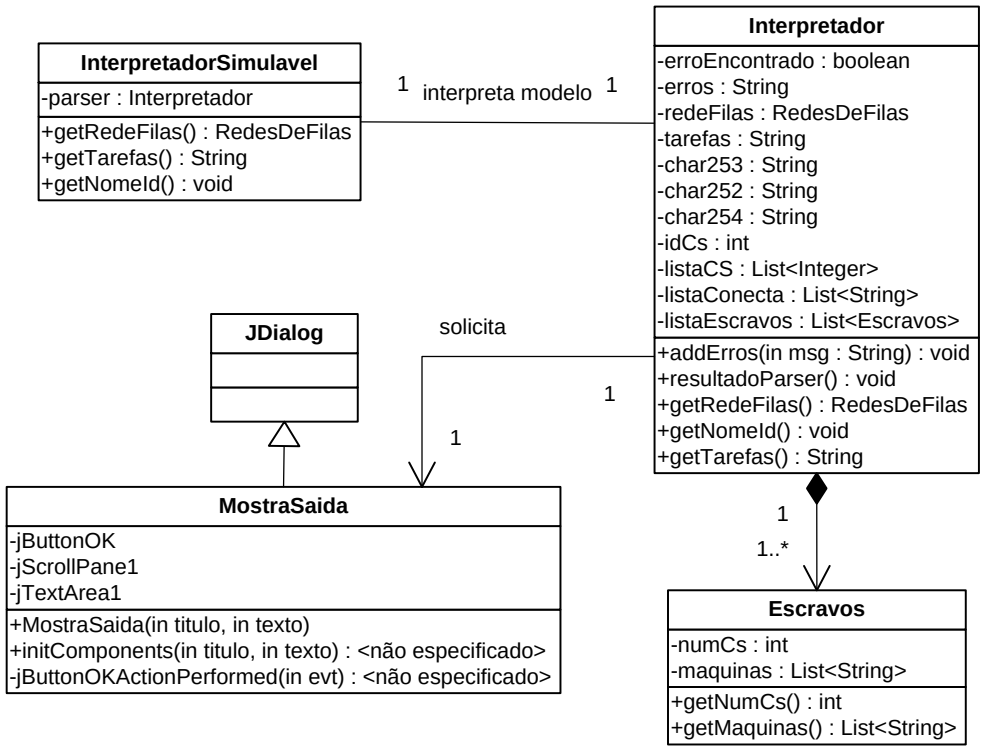

Figura 3.3: Diagrama de classes UML.

3.5.2 Prototipação do Interpretador de Modelos Icônicos

Para realizar a interpretação do modelo icônico para o modelo simulável, o analisador sintático realiza uma leitura do arquivo de especificação do modelo icônico, armazenando os dados em um objeto, que é uma coleção do tipo HashSet da classe DescreveIconesPrivado. Na Figura 3.4 é apresentado o diagrama de classes UML do protótipo do interpretador de modelos icônicos.

Para cada entrada das palavras chaves MAQ, REDE, CLUSTER e INET, é criada uma nova instância da classe DescreveIconePrivado, para onde é passado todos os atributos necessários inseridos pelo usuário para a simulação. Esta instância é adicionada à coleção de objetos do tipo HashSet.

Após todo o arquivo ser reconhecido, é verificado se todos os nomes inseridos são únicos e se todos os nós presentes na lista de escravos do mestre escalonador foram adicionados como um nó na interface.

Se não houver erros nos processos de reconhecimento e validação do modelo icônico, a resposta é retornado para o usuário indicando que o modelo foi reconhecido com sucesso. Em seguida, inicia-se a fase de conversão do modelo icônico para o modelo simulável.

Na conversão do modelo icônico para o modelo simulável, os objetos armazenados dentro da coleção são acessados um a um e seus atributos são organizados e escritos em um arquivo, respeitando as regras da linguagem do modelo simulável.

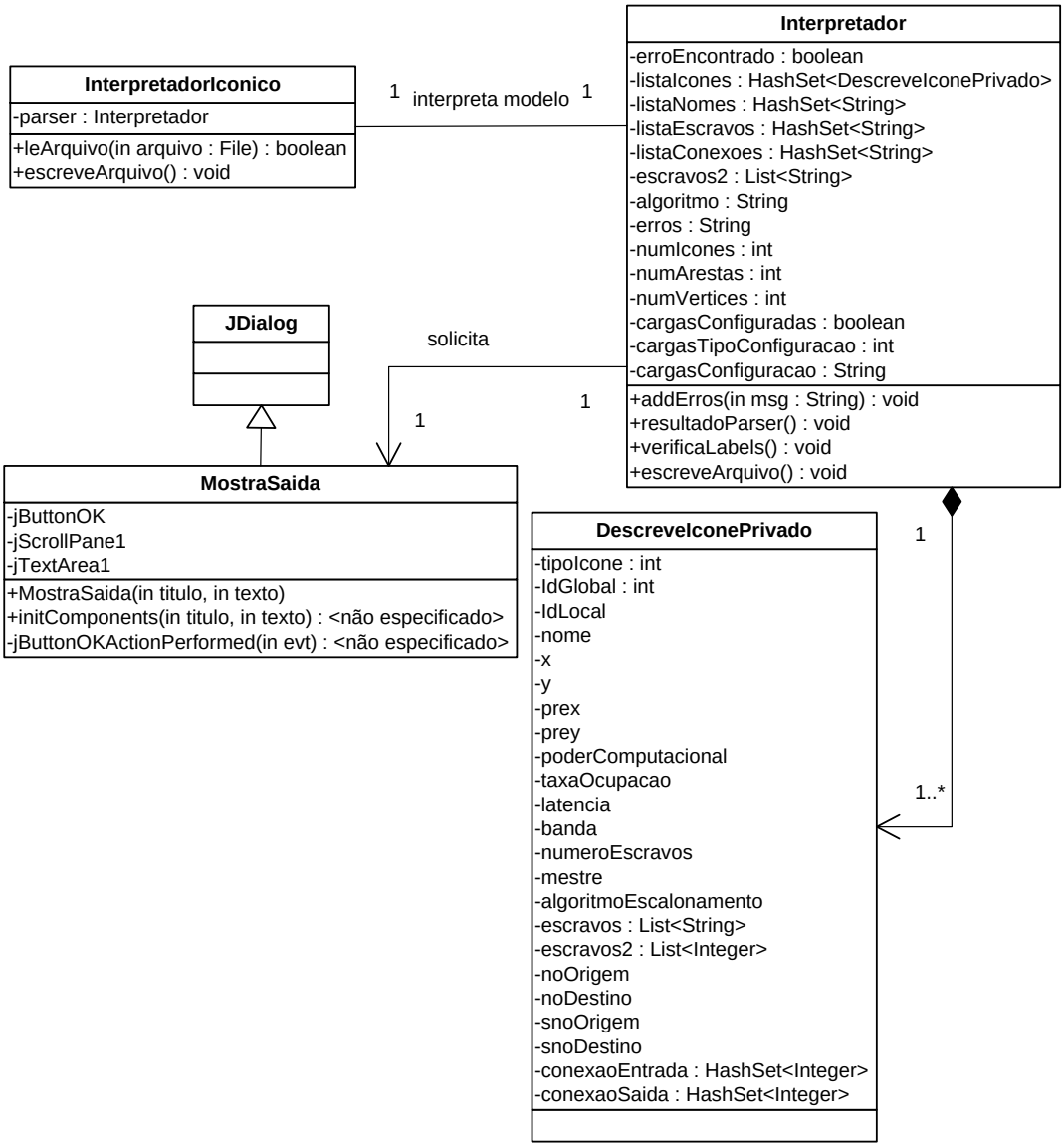

Figura 3.4: Diagrama de classes UML.

3.5.3 Prototipação do Interpretador de Modelos SimGrid

Para realizar a interpretação do modelo externo (SimGrid) para o modelo icônico, o analisador sintático realiza uma leitura do arquivo de especificação do modelo SimGrid, filtra os dados relevantes e estes são armazenados em objetos, que são instâncias das classes estáticas Server, Master, Network e Route. Na Figura 3.5 é apresentado o diagrama de classes UML do protótipo interpretador de modelos SimGrid.

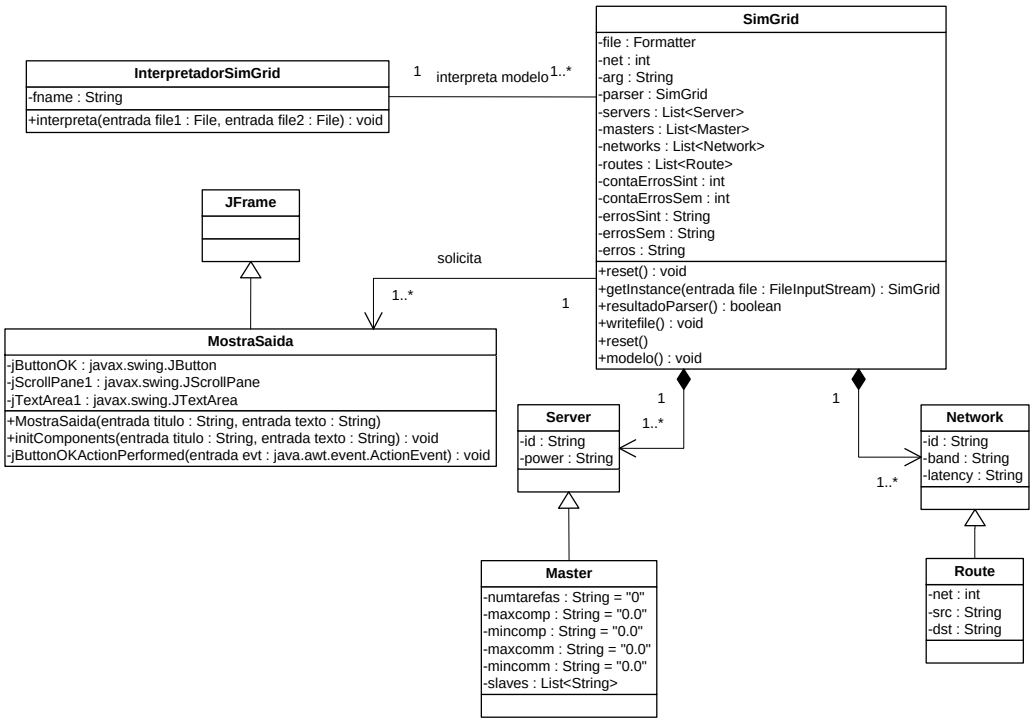

Figura 3.5: Diagrama de classes UML.

A classe Server possui atributos que armazenam o identificador do servidor (id) e o poder computacional (power).

A classe Master é uma subclasse da classe Server e armazena a lista de escravos de cada mestre (slaves), o número de tarefas (numtarefas), valores máximo e mínimo para tamanho das tarefas (maxcomp e mincomp) e comunicação (maxcomm e mincomm) a ser utilizada por cada mestre.

A classe Network armazena o identificador do *link* (id), a largura de banda (band) e latência da rede (latency).

A classe Route é uma subclasse da classe Network e armazena um contador para diferenciar *links* com identificadores iguais (net), a origem (src) e o destino (dst) de cada *link*.

A classe SimGrid armazena todas as classes, os atributos para armazenar o nome do arquivo de saída (File), um contador para o identificador de *link* (net), o atributo para implementar *singleton* (parser), listas de objetos das classes Master (masters), Server (servers), Network (networks) e Route (routes), contadores para erros sintáticos (contaErrosSint) e semânticos (contaeErrosSem) e atributos para armazenar os erros sintáticos (errosSint) e semânticos (errosSem). Além disso, implementa o *design pattern singleton* (através do método getInstance() e do atributo parser) e possui o método writefile() para escrever o modelo gerado no arquivo de saída, o método closefile() para fechá-lo e o método resultadoParser() que verifica se o *parser* encontrou erros e caso tenha encontrado, grava os erros no objeto.

A classe MostraSaida é utilizada apenas para exibir erros encontrados durante o *parser* ao usuário.

A classe InterpretadorSimGrid possui o atributo fname para armazenar o nome do arquivo a ser lido, o método interpreta(File file1, File file2), o método setFileName(entrada f: File) e o método getFileName(). O método interpreta(File file1, File file2) recebe os arquivos application_file.xml e plataform_file.xml como entrada, chama o método getInstance() da classe SimGrid. O método getInstance() serve para criar um objeto SimGrid, caso seja o primeiro modelo a ser interpretador desde a execução do simulador, ou para retornar o objeto da classe SimGrid existente. Após executar o método getInstance(), o método setFileName() é chamado para salvar o nome do arquivo application_file no atributo fname. Em seguida, através do objeto da classe SimGrid, o método ReInit(application_file) é chamado para reinicializar o *parser* caso esta seja a segunda interpretação e chama o método modelo() para realizar o *parser*. Após o processamento do *parser* (leitura dos arquivos de entrada e armazenamento dos dados nos objetos), o nome do arquivo application_file.xml é salvo através do método setFileName(), o método ReInit() reinicia o *parser* novamente para receber o arquivo plataform_file e executar o *parser* novamente. No final do processamento de ambos os arquivos, o método resultadoParser() é chamado para verificar a existência de erros léxicos, sintáticos ou semânticos nos modelos SimGrid. Se nenhum erro for encontrado, o método writefile() gera o modelo icônico

correspondente no arquivo de saída, concluindo a interpretação do modelo externo. Caso contrário, nenhum modelo de saída é gerado e um objeto da classe MostraSaida é instanciado para exibir uma mensagem ao usuário com os erros encontrados durante a interpretação. Por fim, o método reset() é chamado para "limpar" todas as variáveis a fim de preparar o *parser* para receber um novo modelo.

Salienta-se ainda que o método modelo() realiza interface entre as classes privadas e os métodos utilizados pelo interpretador e pelos analisadores léxico e sintático. Estes métodos pertencem à classe SimGrid, no entanto, não foram ilustrados na Figura 3.5 por serem métodos gerados pela ferramenta JavaCC a partir das especificações dos analisadores léxico e sintático. Os métodos *set* e *get* também não foram apresentados neste diagrama de classes. Além disso, o código-fonte Java destes métodos foi gerado pela ferramenta JavaCC a partir das especificações das regras de produção, codificadas em JavaCC, das gramáticas regular e livre de contexto.

Há também outras classes que foram geradas para abstrair os conceitos dos analisadores léxico e sintático a partir da compilação do arquivo com extensão .jj pelo JavaCC e, por este motivo, também não foram ilustradas na Figura 3.5.

Todas essas classes formam o *package* SimGrid, que está contida dentro do *package* InterpretadorExterno do módulo de interpretador de modelos externos. O *package* InterpretadorExterno é importado pelo módulo responsável pela interface com o usuário juntamente com os *packages* do módulo do motor de simulação (Estatistica, NumerosAleatorios, RedesDeFilas e Simulacao), os *packages* do módulo da interface (CarregaArqTexto, Interface e DescreveSistema) e os *packages* ModeloIconico e ModeloSimulavel que estão contidos dentro do *package* do módulo do interpretador interno (InterpretadorInterno).

Na Figura 3.6 é apresentado o diagrama de pacotes UML.

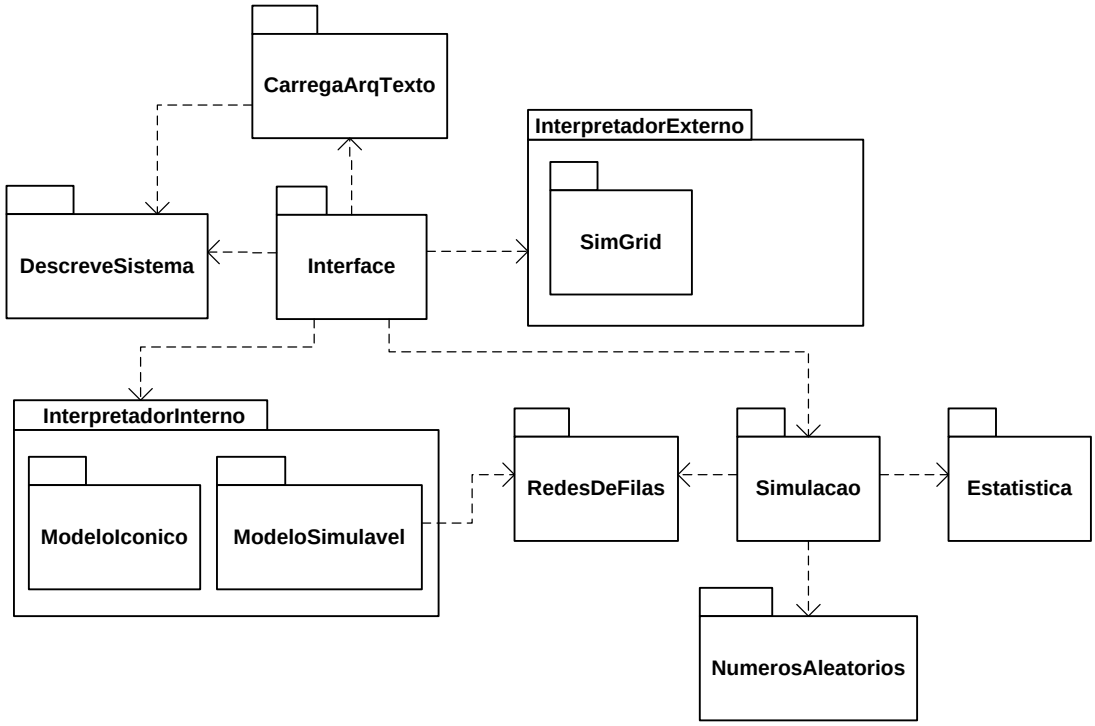

Figura 3.6: Diagrama de pacotes UML.

3.6 Prototipação da Interface Icônica

A interface icônica construída provê recursos ao usuário para que ele tenha condições de efetuar a especificação de um modelo de grade computacional e, com esse modelo, efetuar a simulação de modo que ele obtenha os resultados esperados.

3.6.1 Casos de Uso

Na Figura 3.7 é apresentado o diagrama de casos de uso referente à interface icônica desenvolvida para o simulador. Ao abrir a interface, o usuário tem como opções a elaboração de um novo modelo de grade computacional, ou a possibilidade de abrir um modelo de grade computacional já existente. Tanto para a elaboração de uma nova grade computacional, quanto para a modificação de uma já existente, o usuário tem à sua disposição os recursos fornecidos pela interface, como ícones e parâmetros de componentes de grade computacional.

Após o modelo ser montado e todos os parâmetros serem ajustados, o usuário inicia a simulação. Nessa simulação serão utilizados todos os recursos fornecidos pelo interpretador de modelos icônicos, pelo interpretador de modelos simuláveis e pelo núcleo de simulação (simulador). Após a simulação ser efetuada, os resultados obtidos são retornados com uma análise simples ao usuário, de modo que o usuário possa interpretar claramente o que eles representam.

Na Tabela 3.1 é apresentada a tabela de casos de uso UML para os casos de uso da Figura 3.7.

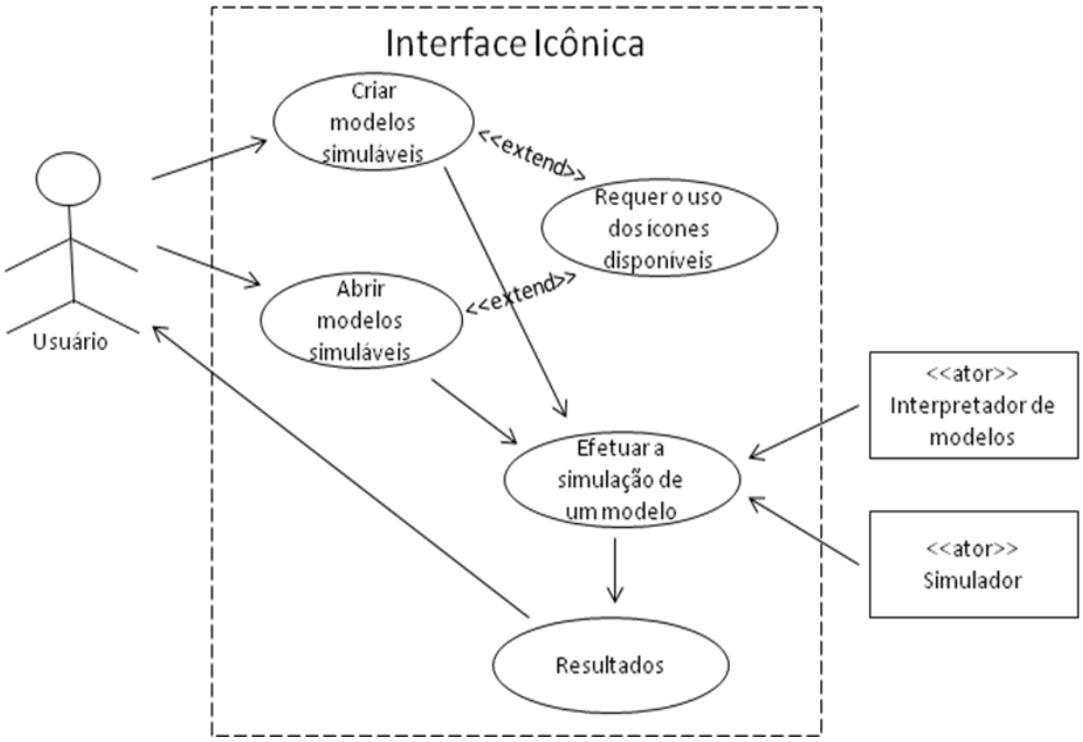

Figura 3.7: Diagrama de casos de uso UML.

Tabela 3.1: Tabela de casos de uso UML.

Caso de uso	Quem inicia a ação	Descrição do caso de uso
Criar modelos simuláveis	Usuário	O usuário acessa a interface para criar um novo modelo a ser simulado.
Abrir modelos simuláveis	Usuário	O usuário acessa a interface para editar um modelo já anteriormente salvo.
Requer o uso dos	---	Extensão para auxiliar a criação,

ícones disponíveis		edição e parametrização do modelo.
Efetuar a simulação de um modelo	Usuário, Simulador e Interpretador de modelos simuláveis.	O usuário após ter criado o modelo ele inicia a simulação do mesmo, que será efetuada pelos atores, interpretador de modelos e simulador.
Resultados	Usuário, Simulador e Interpretador de modelos simuláveis.	Depois de efetuada a simulação os resultados obtidos são entregues ao usuário.

3.6.2 Atividades do Sistema

O sistema provê ao usuário uma interface icônica com opções básicas para elaborar um modelo que possa ser simulado da forma mais realista possível.

Na Figura 3.8 é apresentado o diagrama de atividades UML.

A primeira ação de um novo usuário ao iniciar o uso do sistema, é abrir a tela na qual ele irá inserir os ícones que farão parte do modelo simulável. A ação pode ser feita através da barra de menus, onde se encontra a opção *File->New*, ou ainda, através do atalho *Ctrl+N* do teclado.

Após aberto a tela de desenho, o usuário tem a sua disposição, na barra de ícones identificada por *Icon*, os quatro ícones essenciais que podem ser usados para construir um modelo de grade computacional. Juntamente com os quatro botões que adicionam os ícones na tela, estão o botão que adiciona as tarefas ao sistema e o botão que inicia a simulação. A ação de adicionar um ícone engloba a ação do usuário de selecionar um ícone através dos botões de ícones dispostos e, clicar em uma posição da tela de desenho para que o ícone seja inserido na posição selecionada pelo usuário.

Na Figura 3.9 é apresentada uma visão geral da interface icônica do simulador de grades computacionais.

A tela de desenho permite mobilidade ao usuário para movimentar os ícones conforme desejar e, excluí-los se for preciso. A cada alteração feita pelo usuário na tela de desenho, esta é redesenhada para que as alterações sejam visíveis e, os ícones

são verificados para informar ao usuário se eles estão corretos ou se ainda precisam ser configurados.

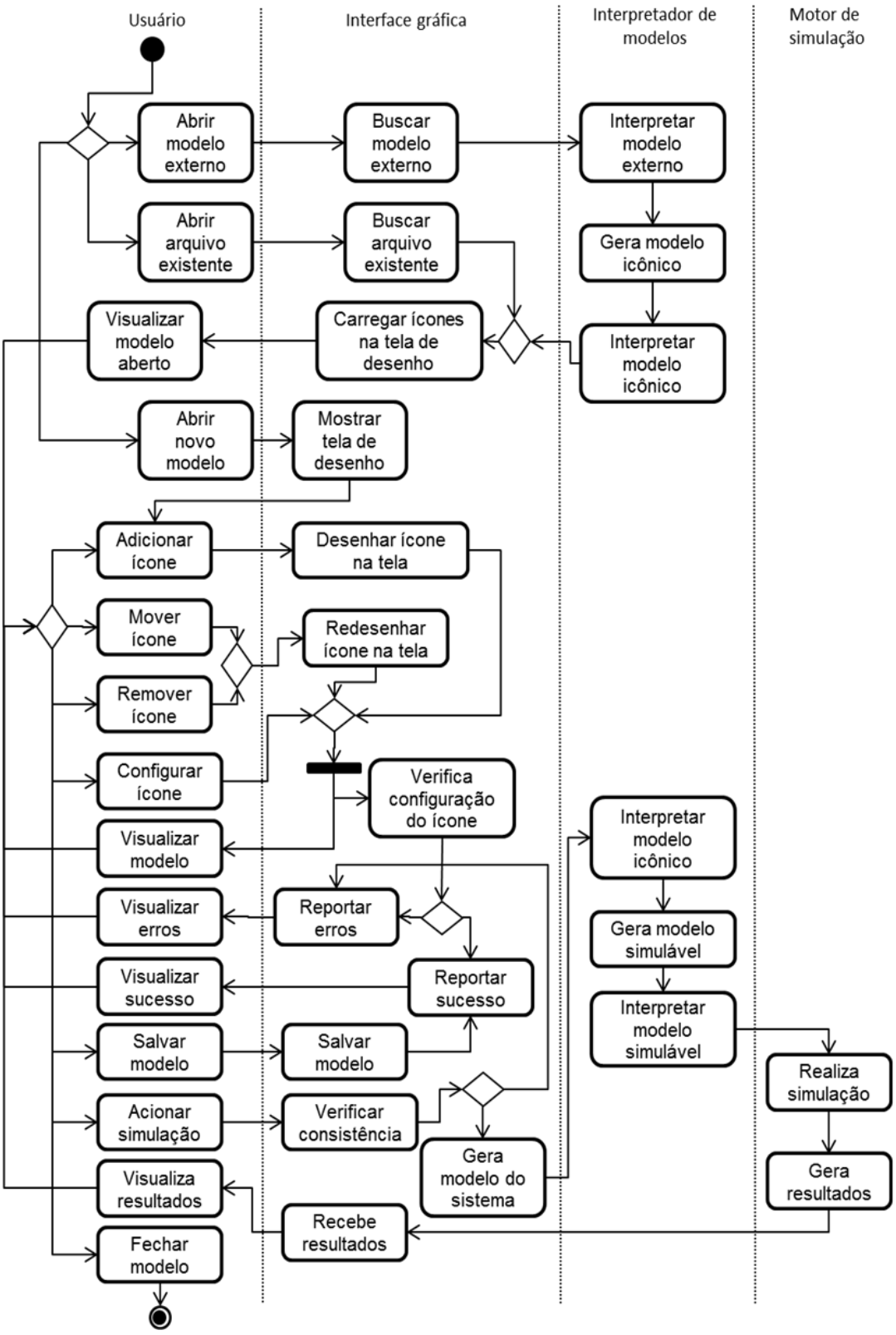

Figura 3.8: Diagrama de atividades UML.

A qualquer momento o usuário pode exercer diferentes atividades no sistema. Ele pode inserir novos ícones, movimentá-los, configurá-los, removê-los e salvar ou fechar o sistema. Além disso, ele poder iniciar a simulação a qualquer momento, desde que o sistema esteja configurado corretamente.

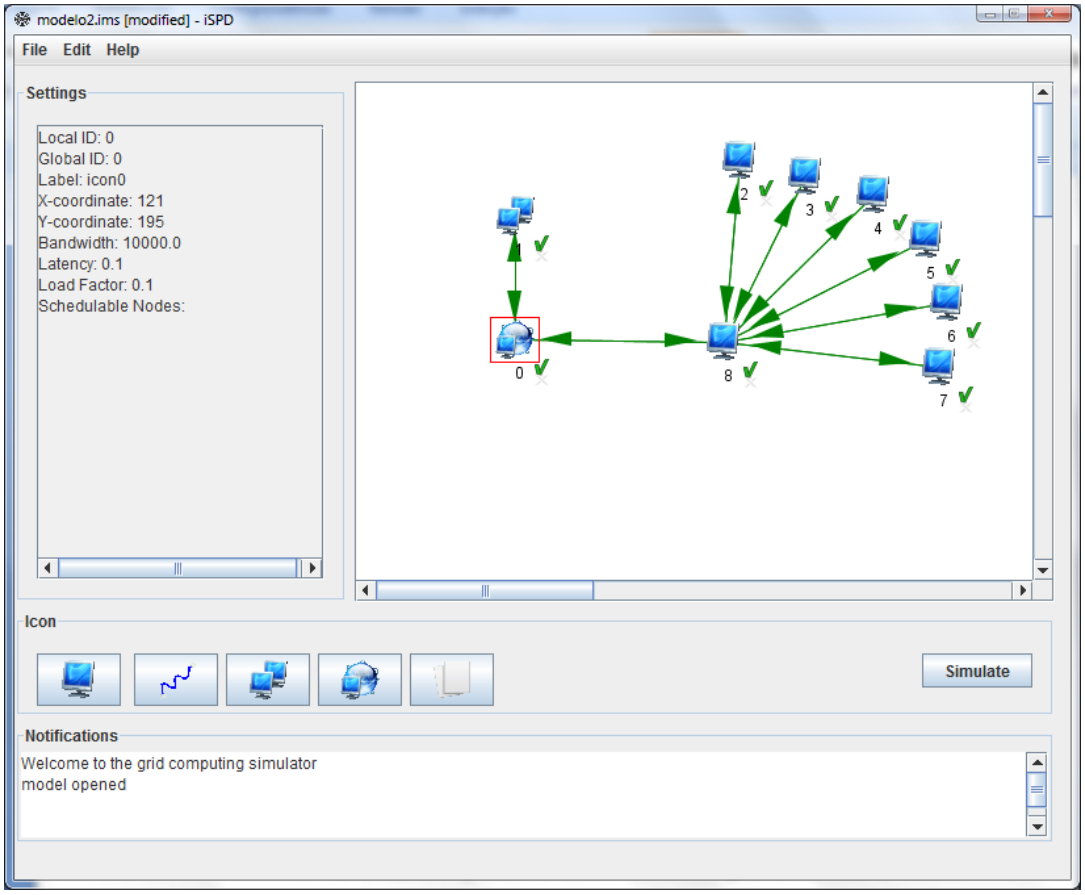

Figura 3.9: Visão geral da interface icônica do simulador.

3.6.3 Estrutura da Interface

A interface possui áreas bem definidas e com funções bem definidas para que o usuário possa aproveitar melhor as funcionalidades do sistema. Todas as áreas do sistema são definidas dentro de uma janela principal, exceto a área onde são inseridos os ícones, pois ela já está definida em uma classe secundária, que é uma instância dentro da janela principal.

A área lateral esquerda da Figura 3.9, denominada *Settings*, é uma área onde podem ser visualizadas todas as configurações para um determinado ícone quando ele é selecionado.

Além disso, foi inserida uma área de notificação para o usuário para que ele saiba tudo a respeito do que está ocorrendo na interface, esta área é denominada *Notifications* e é ilustrada na parte inferior da interface da Figura 3.9. Todas essas funcionalidades servem para que o usuário se sinta bem em utilizar o sistema e seja bem-sucedido em utilizá-lo.

Na Figura 3.10 é apresentado o diagrama de classes UML do protótipo da janela principal.

Ao ser acionado algum dos botões referentes aos ícones da janela principal, a inserção de um novo ícone é informada ao objeto aDesenho e, juntamente com essa informação, é passada a informação de qual ícone será inserido.

Através da operação MouseMoved é possível saber onde o *mouse* está localizado. Quando o usuário pressiona o botão do *mouse* sobre a área de desenho, a operação MouseClicked informa a posição do ponteiro do *mouse* através dos atributos mousex e mousey. Com esses atributos é possível inserir o ícone na posição desejada pelo usuário.

Para que o ícone seja inserido no desenho, é criado um objeto da classe Icone, e esse objeto é inserido em uma estrutura denominada HashSet. A estrutura HashSet é uma classe que implementa a interface Set suportada por uma tabela *hash* (uma instância de HashMap). Esta classe oferece um desempenho constante de tempo para as operações básicas (adicionar e remover), assumindo que a função *hash* dispersa os elementos adequadamente.

Na classe Icone estão os parâmetros referentes aos ícones suportados pelo sistema. A diferenciação é feita através do atributo numIcone do tipo inteiro. Dependendo do seu valor, um determinado ícone será exibido na tela e também todos os seus atributos. As conexões de rede são desenhadas com as funções de desenho drawLine da linguagem java e, para os demais ícones, é utilizada a função drawImage.

Juntamente com o ícone, é desenhado no canto inferior direto, um símbolo que representa a validação das configurações feitas para aquele ícone, o símbolo possui dois estados, um estado de verificado na cor verde representando que as

configurações estão corretas, e um “X” em vermelho, simbolizando que as configurações para aquele ícone estão incorretas.

Figura 3.10: Diagrama de classes do protótipo da janela principal.

Esta funcionalidade permite ao usuário uma fácil visualização e rápida interpretação dos erros cometidos na construção do modelo e, consequentemente, sua fácil correção, trazendo satisfação para o usuário em utilizar o sistema.

Os ícones dispostos na área de desenho podem ser movimentados com o mouse em qualquer direção. Esta ação é feita pela operação MouseDragged, que responde pela movimentação do mouse quando o botão do mouse é acionado.

Uma conexão de rede só pode ser inserida quando a interface já possui, no mínimo, dois ícones inseridos, pois cada extremidade da conexão de rede precisa estar associada a um ícone.

A interface não permite que as conexões de redes sejam arrastadas, uma vez que uma conexão de rede é inserida conectando dois ícones. Ela só pode ser modificada quando há alterações nos ícones de suas extremidades, ou seja, conforme um ícone é movimentado todas as conexões de rede que estão ligadas a ele são movidas também.

3.6.4 Configuração dos Ícones

Os ícones dispostos na tela precisam ser configurados para que o modelo possa ser simulado. Essa configuração é feita através de janelas auxiliares, que são abertas exclusivamente para este fim. Elas podem ser abertas pressionando-se duas vezes o botão do *mouse* sobre o ícone que se deseja configurar.

Na Figura 3.11 é apresentado o diagrama de classes UML dos ícones da interace. Essas classes herdam as propriedades da classe JDialog para que possa ser aberta uma nova janela.

Na Figura 3.12 são apresentadas as janelas de configuração de cada ícone, demonstrando os parâmetros que são atribuídos para cada item.

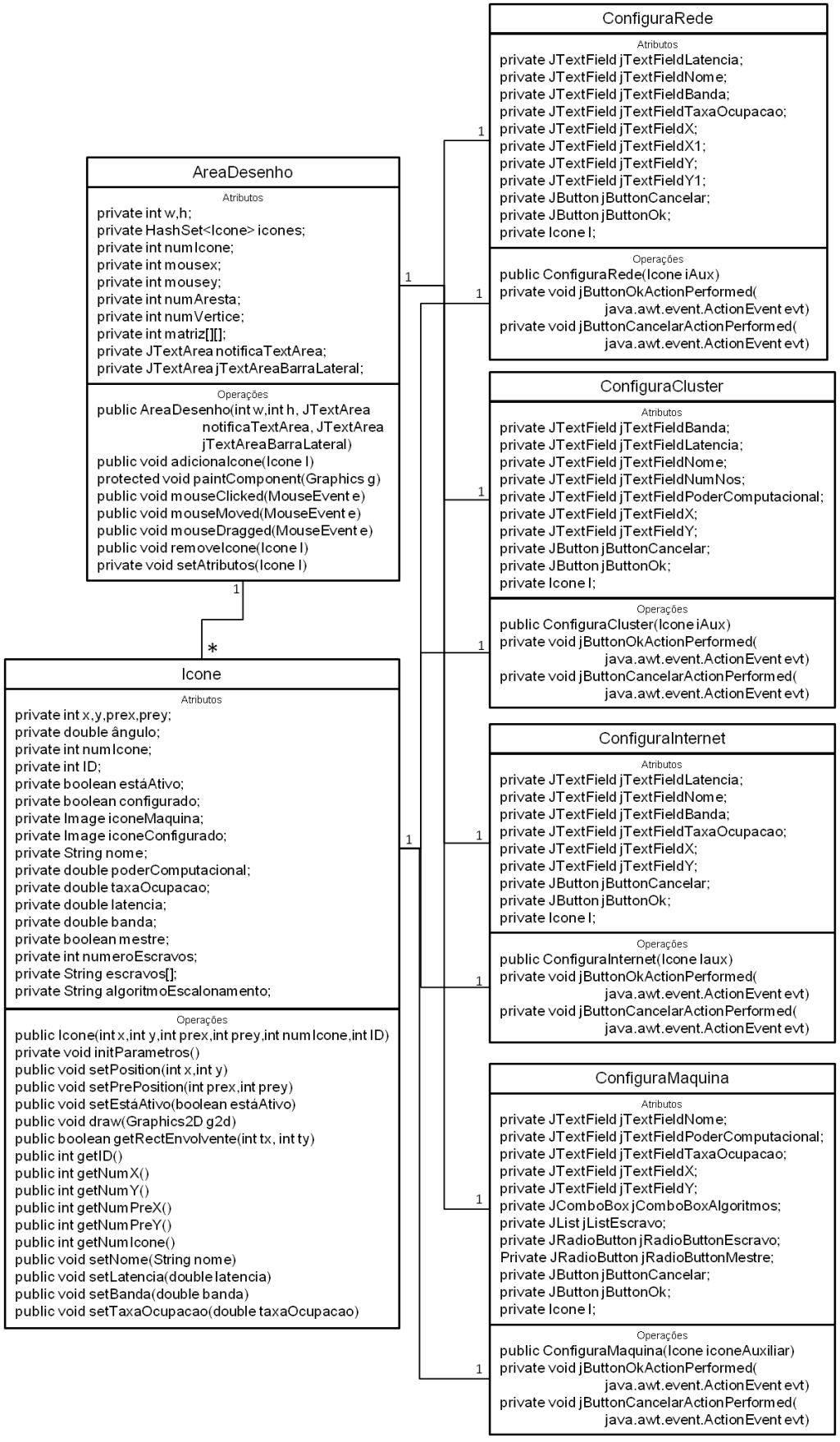

Figura 3.11: Diagrama de classes UML dos ícones da interface.

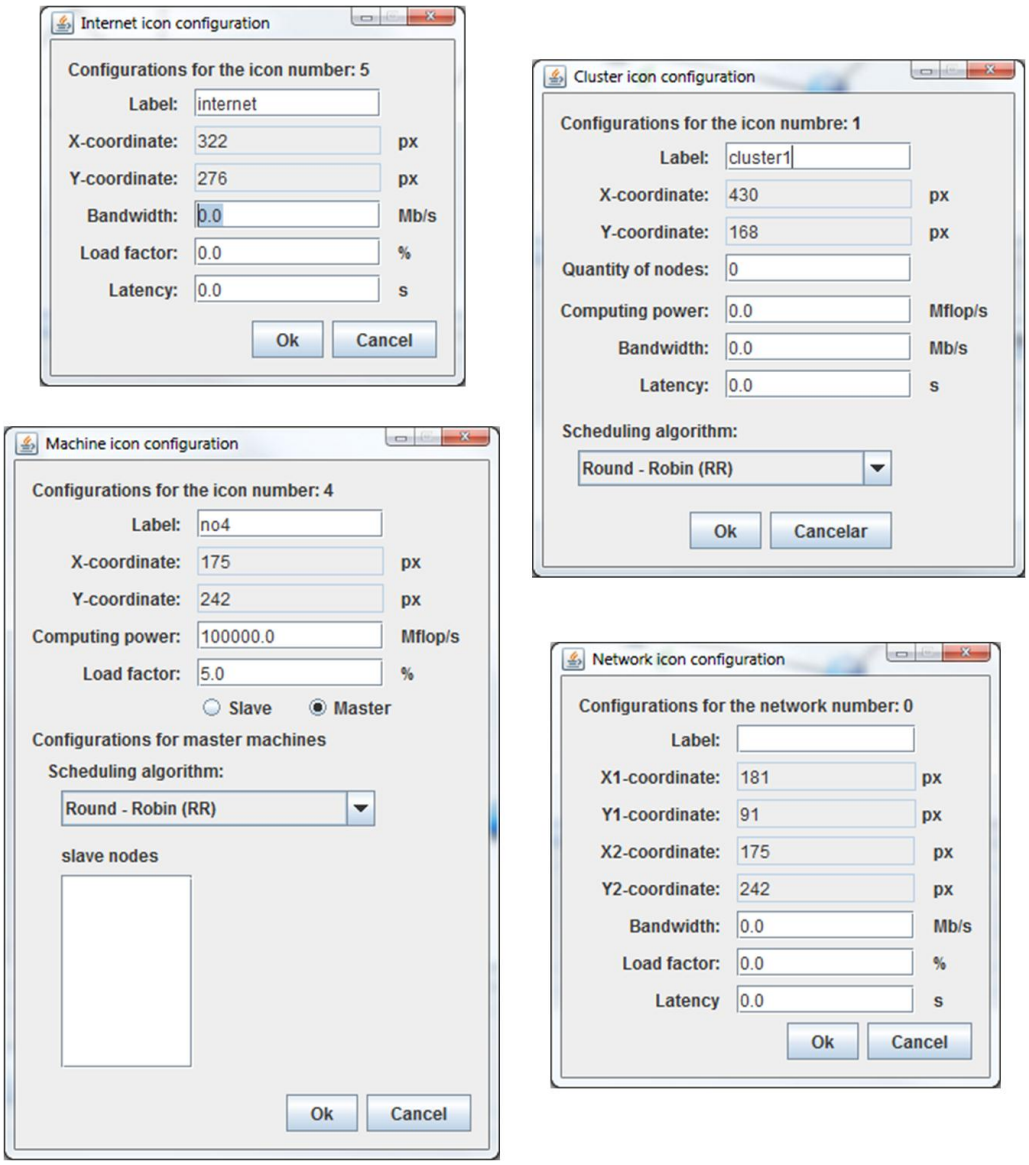

Figura 3.12: Janelas de configuração dos ícones.

3.7 Considerações Finais

Neste capítulo apresentaram-se as gramáticas construídas para as linguagens do modelo simulável, do modelo icônico e de modelos SimGrid e também, as implementações e o funcionamento da interface icônica e dos interpretadores de modelos SimGrid, de modelos simuláveis e de modelos icônicos, juntamente com seus respectivos diagramas UML para ilustrar o sistema.

Capítulo 4 - Testes

4.1 Considerações Iniciais

Neste capítulo serão apresentados todos os testes realizados com o interpretador de modelos SimGrid e com a interface do simulador. Para o interpretador, foram realizados testes de interpretação de modelos livres de erros e testes de modelos com erros léxicos, sintáticos e semânticos. Para a interface, foram realizados testes de funcionalidades dos botões, exibição de menus, ícones e janelas.

4.2 Exemplos de Conversões de Modelo SimGrid para Modelo Icônico

Nesta seção serão exibidos alguns exemplos de modelos icônicos que foram gerados a partir dos modelos de simulação do SimGrid para testar o protótipo de interpretador de modelos SimGrid. Os códigos-fonte completos dos modelos SimGrid construídos podem ser visualizados no Apêndice A, ao final desta monografia.

4.2.1 Exemplo de Conversão 1

A simulação deste exemplo envolve um *cluster* de nove nós (*cluster* do laboratório do GSPD financiado pela FAPESP, processo nº 2008/09312-7) utilizando o algoritmo *Round-Robin* para escalonamento das tarefas. Foram submetidas 20 tarefas a este *cluster*.

Este modelo de simulação necessita de três arquivos de configuração. Um arquivo contém a plataforma a ser utilizada, o segundo inclui as informações da aplicação a ser simulada e o terceiro é um arquivo executável compilado em linguagem C que utiliza as APIs do SimGrid e que implementa o escalonamento das tarefas. Este último não será exibido nesta monografia, pois até o momento estão sendo utilizados algoritmos de escalonamento pré-definidos (*RoundRobin*, *WorkQueue*, FPLTF, etc) e, potanto, não é necessário que o protótipo do interpretador de modelos SimGrid interprete o arquivo escrito em linguagem C.

O arquivo "plataform_file" exemplifica uma plataforma de simulação que contém o poder computacional de cada máquina em Mflops/s, a lista de escravos de cada mestre, a banda da conexão em Mb/s, a latência da rede e as rotas de ligação dos nós.

Já o arquivo "application_file" define como cada *host* será identificado e qual função ele irá exercer (mestre ou escravo). Os escravos não necessitam de nenhum atributo, já os mestres precisam de parâmetros como, por exemplo, número de tarefas que serão executadas, o tamanho de cada uma e quanto de comunicação cada *host* utilizará.

Na Figura 4.1 é apresentada a plataforma do modelo SimGrid convertido para o modelo icônico, onde uma máquina mestre (*node* 0 - gspd-fe) possui poder computacional de 10000.0 Mflop/s, taxa de ocupação de 0% (servidor ocioso) e oito escravos (*node* 1 - gspd-node1, *node* 2 - gspd-node2, *node* 3 - gspd-node3, *node* 4 - gspd-node4, *node* 5 - gspd-node5, node 6 - gspd-node6, node 7 - gspd-node7 e *node* 8 - gspd-node8), todos com 10000.0 Mflop/s de poder computacional e 0% de taxa de ocupação.

Em seguida, são exibidos os *links*, cada um com 100.0 Mb/s de banda, 0.001 s de latência, 0% de taxa de ocupação (rede ociosa) e as máquinas conectadas através do *link*.

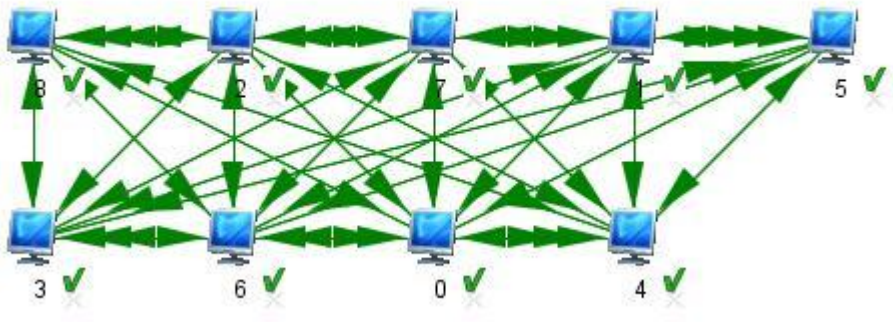

Figura 4.1: Exemplo 1 de plataforma do modelo icônico.

No modelo SimGrid, existem apenas *links half-duplex* e eles podem possuir o mesmo identificador. Quando é necessário representar um *link full-duplex*, utilizam-se dois *links half-duplex* com o mesmo identificador. No modelo icônico, um *link* pode ser *half-duplex* ou *full-duplex*, mas possui um identificador único. Portanto, é necessário atribuir um índice ao identificador (link[id]) quando a intenção é representar um *link full-duplex* do SimGrid, já que nele haverão dois *links* com o mesmo identificador.

Por último, é definida a carga de trabalho que é do tipo "máquina", o identificador da máquina mestre (gspd-fe) que recebe a carga de trabalho, número de tarefas (20), tamanho máximo da tarefa (20000.0) e tamanho máximo de comunicação (20.0). Não foram definidos valores de tamanho mínimo para tarefa e comunicação (default = 0.0). Além disso, a forma de distribuição da carga de trabalho é a Poisson.

4.2.2 Exemplo de Conversão 2

Na Figura 4.2 é apresentada a plataforma de outro modelo SimGrid convertido em modelo icônico, onde uma máquina mestre (gspd-fe) possui poder computacional de 20000.0 Mflop/s, taxa de ocupação de 0% (servidor ocioso) e

dezoito escravos (gspd-node1, gspd-node2, gspd-node3, gspd-node4, gspd-node5, gspd-node6, gspd-node7, gspd-node8, gspd-node9, gspd-node10, gspd-node11, gspd-node12, gspd-node13, gspd-node14 e gspd-node15), todos com 20000.0 Mflop/s de poder computacional e 0% de taxa de ocupação.

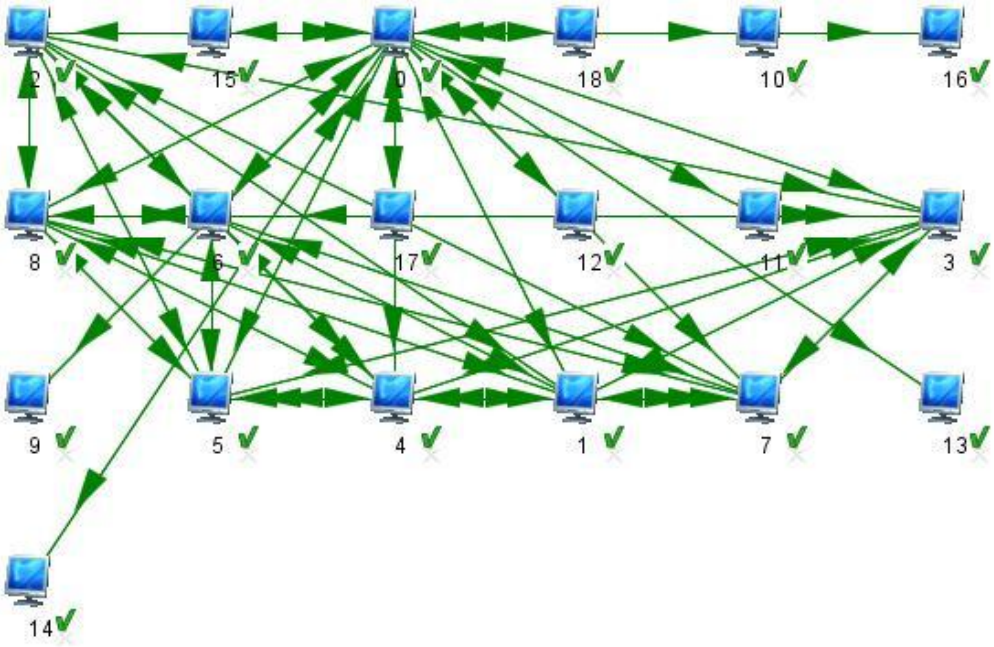

Figura 4.2: Exemplo 2 de plataforma do modelo icônico.

Em seguida, são exibidos os *links*, cada um com 200.0 Mb/s de banda, 0.0001 s de latência, 0% de taxa de ocupação (rede ociosa) e as máquinas conectadas através do *link*.

Por último, é definida a carga de trabalho que é do tipo “máquina”, o identificador da máquina mestre (gspd-fe) que recebe a carga de trabalho, número de tarefas (20), tamanho máximo da tarefa (15000.0) e tamanho máximo de comunicação (50.0). Não foram definidos valores de tamanho mínimo para tarefa e comunicação (default = 0.0). Além disso, a forma de distribuição da carga de trabalho é a Poisson.

4.2.3 Exemplo de Conversão com Erros

Na Figura 4.3 são apresentadas mensagens de erros léxicos, sintáticos e semânticos que correspondem aos erros encontrados no modelo SimGrid importado.

Os erros léxicos deste modelo incluem: identificador “1gspd-node1” começando com um dígito e identificador gspdç-de contém caracter inválido (“ç”).

O erro sintático deste modelo inclui: sinal de “>” não fechado na linha 17.

Os erros semânticos deste modelo incluem: identificador “gspd-node5” não declarado e identificador “gspd-node2” declarado mais de uma vez.

Estes erros fazem com que o modelo icônico não seja gerado e a mensagem a seguir é exibida para o usuário.

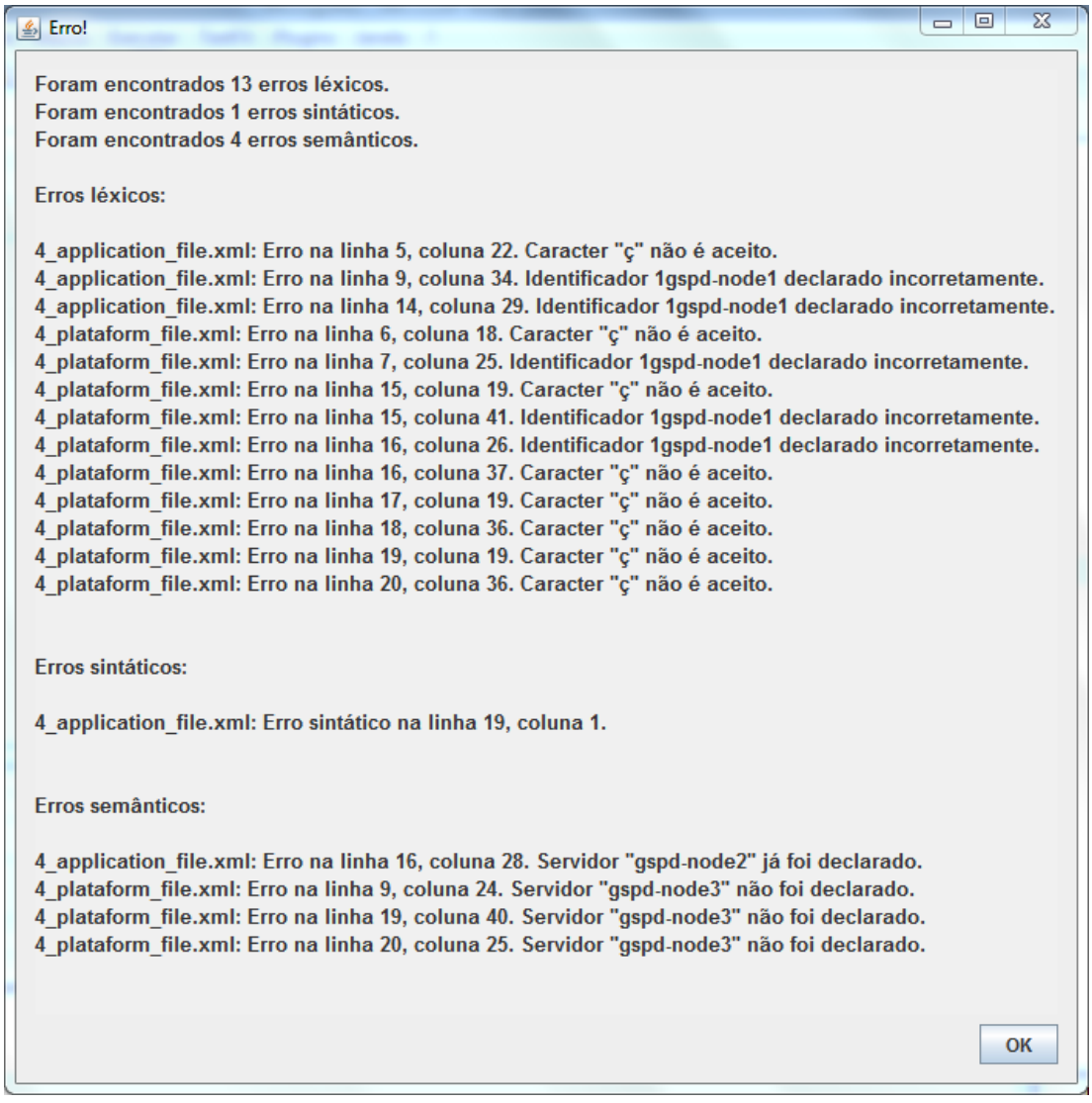

Figura 4.3: Exemplos de erros léxicos.

4.3 Uso da Interface Icônica

A interface icônica proporciona facilidade ao usuário através da grade e da régua de desenho. Estes recursos são opcionais para o usuário, ou seja, ele pode ser inserido ou removido a qualquer momento através da barra de menus nas opções *Edit ->Drawing Grid* (*Ctrl+G*) ou *Edit->Drawing Rule* (*Ctrl+R*). Além disso, estes recursos facilitam o posicionamento dos ícones para o usuário, de modo que aumenta a visibilidade da distância entre um ícone e outro.

Na Figura 4.4 é apresentada a interface com a grade e a régua habilitadas.

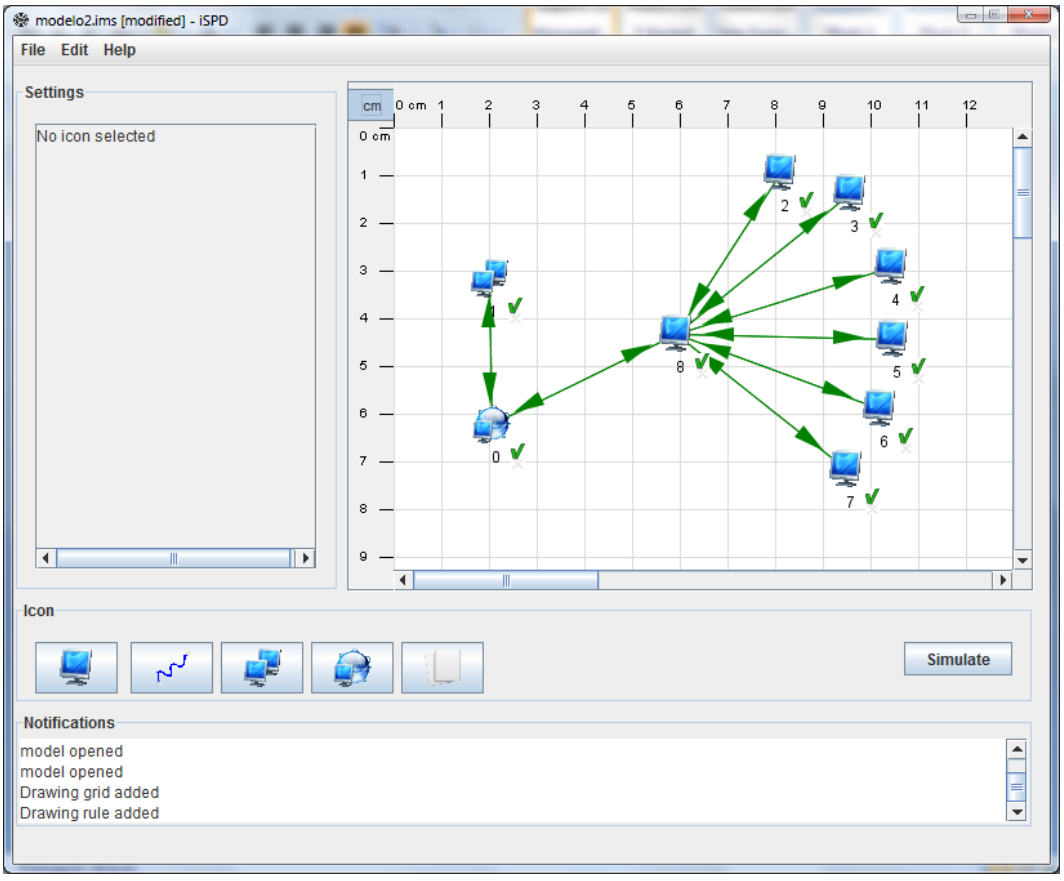

Figura 4.4: Grade e régua da interface icônica.

As mensagens de erros mostram para o usuário quais ações não podem ser realizadas, e como realizá-la de forma correta.

Caso o usuário tente inserir uma conexão de rede em algum lugar que não tenha um ícone, será exibido um aviso. Na Figura 4.5 é apresentado este aviso.

Figura 4.5: Figura ilustrando mensagem de erro ao inserir ícone de rede.

Caso algum parâmetro de algum ícone não esteja configurado, o ícone possuirá um indicador, mostando um “X” em vermelho no canto do ícone, para indicar que não está configurado corretamente. Na Figura 4.6 é apresentado um exemplo desta situação.

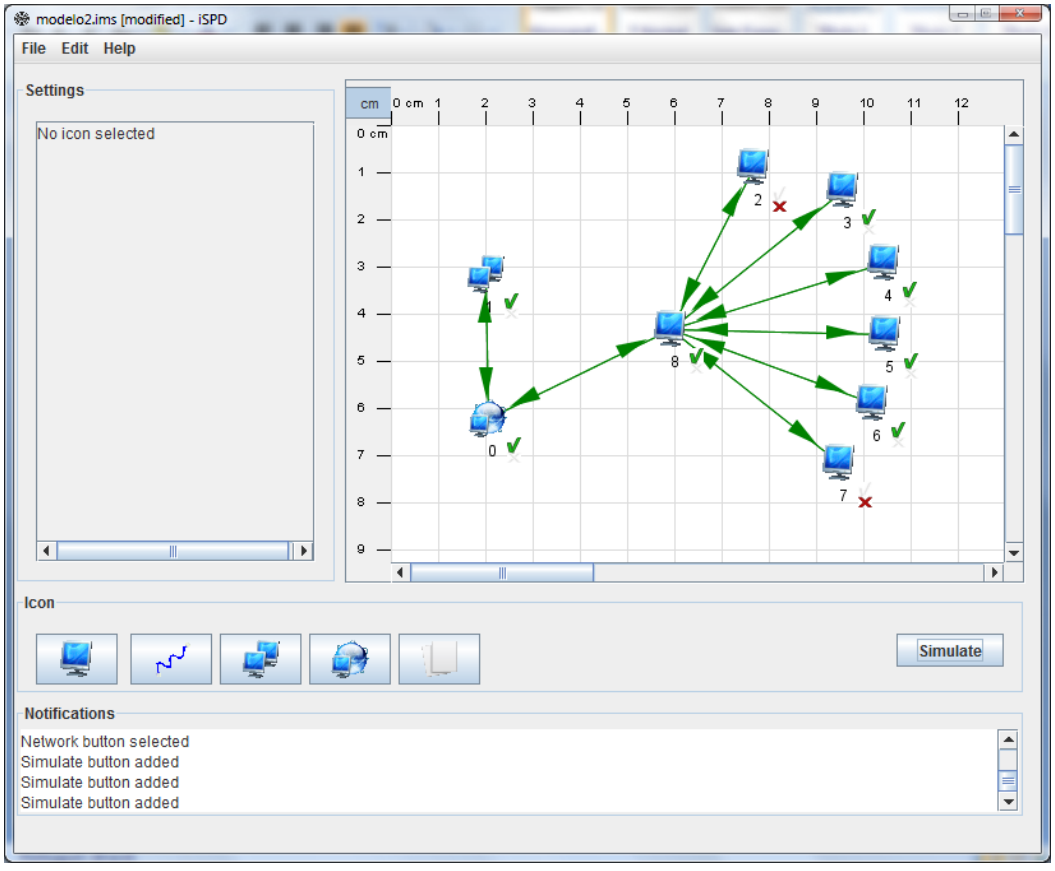

Figura 4.6: Ícone não configurado corretamente.

Quando o usuário acionar o botão *Simulate*, é verificado se todos os parâmetros estão configurados corretamente. Caso algum parâmetro não esteja

configurado, é exibida uma mensagem de erro para o usuário. Na Figura 4.7 é apresentada esta mensagem.

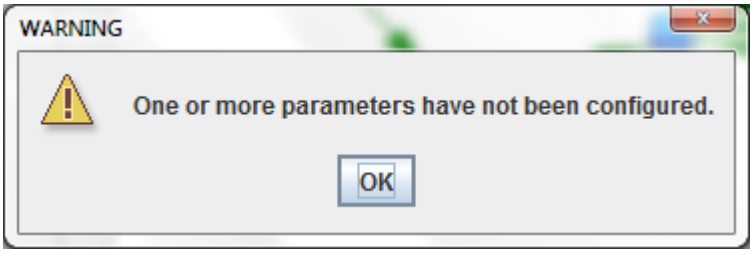

Figura 4.7: Alguns ícones não foram configurados.

Quando todos os ícones estiverem configurados, é necessário que os parâmetros para tarefa também sejam configurados. Essa verificação é realizada antes de iniciar a simulação. Caso as tarefas não tenham sido configuradas, é exibida uma mensagem de erro para o usuário. Na Figura 4.8 é apresentada esta mensagem.

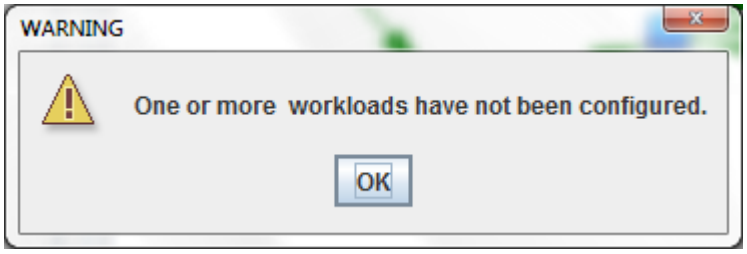

Figura 4.8: Algumas tarefas não foram configuradas.

Quando todos os parâmetros estiverem configurados corretamente e as tarefas tiverem sido escolhidas, a simulação pode ser iniciada. Para dar inicio à simulação, um arquivo texto é gerado contendo as descrições do sistema no formato da linguagem do modelo icônico. Esse arquivo texto é lido pelo interpretador de modelos icônicos e converte toda a descrição do sistema para outro arquivo, agora seguindo as definições especificadas na linguagem do modelo simulável. Esse novo arquivo será lido pelo interpretador de modelos simuláveis e então, convertido em um objeto da classe RedeDeFila, o qual será passado para o simulador para efetuar a simulação. No Apêndice B é apresentado um exemplo de modelo do sistema e no Apêndice C é apresentado o modelo simulável correspondente.

O modelo icônico e o modelo simulável não são acessíveis ao usuário, eles são usados exclusivamente para a comunicação entre a interface icônica e o motor de simulação. Estes modelos são gerados nos momentos apropriados para sua utilização e excluídos assim que desnecessários. Essa comunicação é facilitada com o uso destes modelos, pois quando houver alteração na interface icônica, não será preciso alterar o motor de simulação, ou vice-versa.

Como os arquivos de saída do interpretador de modelo icônicos e do interpretador de modelos simuláveis são gerados dentro do próprio sistema, eles nunca terão erros, pois a validação dos parâmetros de entrada fornecidos pelo usuário é realizada antes da interpretação dos modelos, ou seja, logo que o usuário indicar ao sistema que já terminou de inserir os parâmetros necessários.

Um dos reconhecimentos realizado na interface icônica para evitar que erros sejam propagados para os interpretadores é, por exemplo, a validação dos nomes dos ícones (os nomes precisam começar exclusivamente com uma letra e todos os caracteres devem pertencer a gramática aceita pelo interpretador). Quando o usuário tentar inserir um nome que não seja válido, será exibida uma mensagem de erro. Na Figura 4.9 é apresentada esta mensagem.

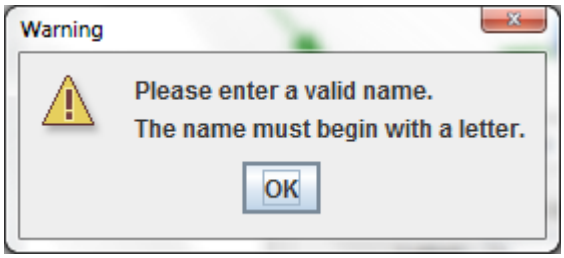

Figura 4.9: Nome incorreto.

Outra validação feita pela interface é a validação dos campos numéricos como, por exemplo, valores inteiros e valores reais.

Na Figura 4.10 é apresentada uma mensagem de erro indicando que deve ser inserido um valor inteiro.

Na Figura 4.11 é apresentada uma mensagem de erro indicando a forma correta de se inserir um número real.

Figura 4.10: Solicitação para entrar com valor inteiro.

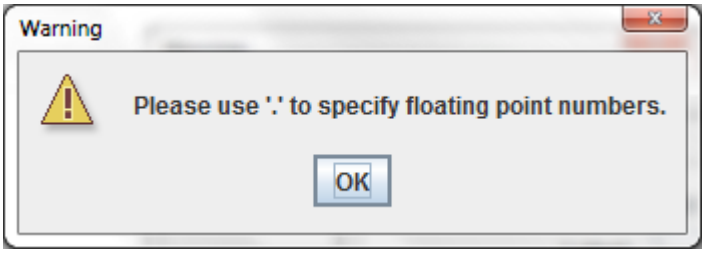

Figura 4.11: Formato correto de um número real.

4.4 Considerações Finais

Nesta seção foram exibidos alguns testes realizados com o interpretador de modelos SimGrid através de exemplos de conversões de modelos SimGrid sem erros e modelos com erros léxicos, sintáticos e semânticos.

Os testes do interpretador de modelos icônicos e do interpretador de modelos simuláveis foram realizados juntamente com os testes da interface, visto que as validações dos modelos icônicos e dos modelos simuláveis são realizadas durante a entrada de parâmetros pelo usuário na interface icônica.

Por fim, a interface icônica foi testada e validada através dos testes das mensagens de erros e de testes com os modelos dos Apêndices B e C.

Capítulo 5 – Conclusões

5.1 Contribuições e Dificuldades Encontradas

A elaboração deste trabalho permite que usuários com pouco conhecimento sobre simulação de grades computacionais sejam capazes de construir uma plataforma de simulação ou importar modelos de simulação do SimGrid de forma fácil e intuitiva, visto que a ferramenta construída possui uma interface icônica amigável. Além disso, as mensagens de erro geradas pelos protótipos dos interpretadores do simulador são de fácil entendimento, dessa forma o usuário pode corrigir seus modelos sem muitas dificuldades.

As principais dificuldades encontradas durante o desenvolvimento deste trabalho foram a identificação da gramática da linguagem utilizada pelo simulador SimGrid e o aprendizado da linguagem Java para implementar todo o sistema (principalmente a interface) e da ferramenta JavaCC para construir os protótipos do interpretador interno e do interpretador de modelos SimGrid.

Este projeto foi publicado e apresentado em dois eventos do estado de São Paulo a fim de expor os resultados obtidos com o desenvolvimento deste trabalho, são eles:

- XXI Congresso de Iniciação Científica da Unesp (CIC Unesp) de São José do Rio Preto - SP [Aoqui *et al*., (2009)] [Guerra *et al*., (2009)] [Oliveira *et al*., (2009)];
- I Escola Regional de Alto Desempenho de São Paulo (ERAD-SP 2010) [Aoqui *et al*., (2010)] [Guerra *et al*., (2010)] [Oliveira *et al*., (2010)].

5.2 Conclusões

O estudo dos principais simuladores existentes atualmente possibilitou contribuir com a identificação de requisitos funcionais e métricas de desempenho necessárias para um ambiente flexível e simples de usar. Dessa forma, a partir do levantamento de requisitos, do estudo sobre redes de filas e do estudo sobre projeto de interface, foi possível realizar a especificação das gramáticas para as linguagens do modelo simulável, do modelo icônico e do modelo SimGrid e em paralelo, projetar e implementar um protótipo de uma interface icônica para o simulador utilizando a linguagem Java.

Com as gramáticas definidas e a interface implementada, utilizou-se a ferramenta JavaCC para construir os analisadores léxico e sintático para as gramáticas do modelo simulável, do modelo icônico e de modelos SimGrid e ainda, implementar e testar os protótipos dos interpretadores do modelo simulável, do modelo icônico e de modelos SimGrid.

A interface criada a partir de todo estudo realizado permite ao usuário criar, de maneira fácil e rápida, um modelo simulável. Essa facilidade e rapidez trazem a satisfação em utilizar a interface, além dos recursos incluídos para mostrar ao usuário respostas claras e objetivas.

Com isso, todas as atividades citadas no cronograma da proposta de projeto final foram cumpridas fielmente.

5.3 Propostas Para Trabalhos Futuros

Para trabalhos futuros, espera-se que o interpretador de modelos externos seja capaz de reconhecer modelos de outros simuladores além do SimGrid como, por exemplo, GridSim e OptorSim, e que a interface adquira novos recursos para facilitar o trabalho do usuário, tais como, os ícones que serão necessários para atender os requisitos dos simuladores GridSim e OptorSim, novos ícones para construir a plataforma de simulação e ícones de atalhos para acessar funcionalidades da ferramenta, possibilitando ao usuário montar uma plataforma de simulação a partir dos ícones ou importar modelos de outros simuladores a fim de gerar automaticamente o modelo icônico correspondente.

Referências Bibliográficas

[Aoqui *et al*., (2009)] AOQUI, V., BRAIT, T. P., GUERRA, A. I., OLIVEIRA, P. H. M. A. e LOBATO, R. S. **Interpretador de Modelos Multiformato Para Plataforma de Simulação de Grades Computacionais**. In: XXI Congresso de Iniciação Científica da Unesp. Anais do XXI Congresso de Iniciação Científica da UNESP. v. CD-ROM. p. 1-4. São José do Rio Preto: Instituto de Biociências, Letras e Ciências Exatas – IBILCE-UNESP, 2009;

[Aoqui *et al*., (2010)] AOQUI, V., GUERRA, A. I., GARCIA, M. A. B. A., OLIVEIRA, P. H. M. A., LOBATO, R. S. e MANACERO JR, A. **Interpretador de Modelos Externos Para Simulador de Grades Computacionais**. In: I Escola Regional de Alto Desempenho de São Paulo. v. CD-ROM. p. 1-2. São Paulo: Universidade Presbiteriana Mackenzie, 2010;

[Bell *et al.*, (2003)] BELL, W., CAMERON, D., G., CAPOZZA, L., MILLAR, A., P., STOCKINGER, K. e ZINI, F., **Optorsim: a grid simulator for studying**

dynamic data replication strategies. International Journal of High Performance Computing Applications, p. 403–416, 2003;

[Casanova, (2001)] CASANOVA, H., **Simgrid: A toolkit for the simulation of application scheduling**. In: 1st International Symposium on Cluster Computing and the Grid (CCGrid 2001). [s.n.], 2001;

[Coulouris *et al.*, (2005)] COULOURIS, G., DOLLIMORE, J. e KINDBERG, T., **Distributed Systems: Concepts and Design**. 4ª ed, Harlow: Addison-Wesley, 2005;

[De Souza *et al.*, (1999)] DE SOUZA, C. S., LEITE, J. C., PRATES, R. O. e BARBOSA, S. D. J.**, Projeto de Interfaces de Usuário: Perspectivas Cognitiva e Semiótica**, Anais da Jornada de Atualização em Informática, XIX Congresso da Sociedade Brasileira de Computação, Rio de Janeiro, 1999;

[Deitel e Deitel, (2006)] DEITEL, H. M., DEITEL, P. J., **Java how to program**. 7 ed. Upper Saddle River: Prentice Hall, 2006;

[Delamaro, (2004)] DELAMARO, M. E., **Como Construir Um Compilador Utilizando Ferramentas Java**, São Paulo: Novatec, 2004;

[Dix *et al.*, (2004)] DIX, A., FINLAY, J., ABOWD, G. D. e BEALE, R., **Human-Computer Interaction**, Ed.3: Prentice Hall, 2004;

[EU DataGrid Project, (2001)] EU DataGrid Project, **The DataGrid architecture**. Technical Report DataGrid-12-D12.4-333671-3-0,C E R N ,Geneva, Switzerland, 2001;

[Falavinha Jr., (2006)(1)] FALAVINHA Jr., J.N., **Avaliação de algoritmos de escalonamento para tarefas em Grids**. Estudos Especiais II. Faculdade de Engenharia - UNESP - Campus de Ilha Solteira Setembro/2006;

[Falavinha Jr., (2006)(2)] FALAVINHA Jr., J.N., **Particionamento e Escalonamento em Grids**. Estudos Especiais I. Faculdade de Engenharia - UNESP - Campus de Ilha Solteira Maio/2006;

[Guerra *et al*., (2009)] GUERRA, A. I., AOQUI, V., GARCIA, M. A. B. A., OLIVEIRA, P. H. M. A. e LOBATO, R. S. **Projeto de Interface Icônica Para Simulador de Grades Computacionais**. In: XXI Congresso de Iniciação Científica da Unesp. Anais do XXI Congresso de Iniciação Científica da UNESP. v. CD-ROM. p. 1-4. São José do Rio Preto: Instituto de Biociências, Letras e Ciências Exatas – IBILCE-UNESP, 2009;

[Guerra *et al*., (2010)] GUERRA, A. I., GARCIA, M. A B. A., OLIVEIRA, P. H. M. A., AOQUI, V., LOBATO, R. S. e MANACERO JR, A. **Plataforma de Simulação de Grades Computacionais: Interface Icônica**. In: I Escola Regional de Alto Desempenho de São Paulo. v. CD-ROM. p. 1-2. São Paulo: Universidade Presbiteriana Mackenzie, 2010;

[GRAS, (1999)] **Module GRAS do SimGrid**. Disponível em: <http://simgrid.gforge.inria.fr/doc/group__GRAS__API.html>. Acesso em 19 de setembro de 2010;

[Java, (2010)] **Java**. Disponível em: <http://www.java.com/pt_BR>. Acesso em: 11 out. 2010;

[JavaCC, (2010)] **JavaCC**. Disponível em: <https://javacc.dev.java.net>. Acesso em: 11 out. 2010;

[JDK, (2010)] **Java Development Kit**. Disponível em: <http://java.sun.com/javase/downloads/widget/jdk6.jsp>. Acesso em: 11 out. 2010;

[JJDoc, (2010)] **JJDoc**. Disponível em: <https://javacc.dev.java.net/doc/JJDoc.html>. Acesso em: 11 out. 2010;

[JJTree, (2010)] **JJTree**. Disponível em: <https://javacc.dev.java.net/doc/JJTree.html>. Acesso em: 11 out. 2010;

[MSG, (1999)] **Module MSG do SimGrid**. Disponível em: <http://simgrid.gforge.inria.fr/doc/group__MSG__API.html>. Acesso em 19 de setembro de 2010;

[Oliveira, (2007)] OLIVEIRA, L. J., **Comparação de ferramentas de simulação de grades computacionais**. Dissertação de Mestrado em Ciência da Computação. São José do Rio Preto: Instituto de Biociências, Letras e Ciências Exatas – IBILCE-UNESP, 2007;

[Oliveira, (2008)] OLIVEIRA, L., J., **Comparação de ferramentas de simulação de grades computacionais**. Relatório Técnico DCCE/IBILCE/UNESP, 2008. Disponível em: <http://www.dcce.ibilce.unesp.br/spd/pubs/gridsimulationtools.pdf>;

[Oliveira *et al*., (2009)] OLIVEIRA, P. H. M. A., AOQUI, V., GUERRA, A. I., GARCIA, M. A. B. A. e MANACERO JR, A. **Especificação de Um Motor de Simulação de Grades Computacionais**. In: XXI Congresso de Iniciação Científica da Unesp. Anais do XXI Congresso de Iniciação Científica da UNESP. v. CD-ROM. p. 1-4. São José do Rio Preto: Instituto de Biociências, Letras e Ciências Exatas – IBILCE-UNESP, 2009;

[Oliveira *et al*., (2010)] OLIVEIRA, P. H. M. A., GUERRA, A. I., GARCIA, M. A. B. A., AOQUI, V., MANACERO JR, A. e LOBATO, R. S. **O Motor de Uma Plataforma de Simulação de Grades Computacionais**. In: I Escola Regional de Alto Desempenho de São Paulo. v. CD-ROM. p. 1-2. São Paulo: Universidade Presbiteriana Mackenzie, 2010;

[Reis, (2005)] REIS, V. Q., **Estudo em grids computacionais: estudo de caso**. Dissertação (Mestrado em Ciências de Computação e Matemática

Computacional). São Carlos: Instituto de Ciências Matemáticas e de Computação – USP, 2005;

[Sharp *et al.*, (2007)] SHARP, H., ROGERS, Y. e PREECE, J., **Interaction design: beyond human-computer interaction**, Ed.2, Wiley, 2007;

[SimDag, (1999)] **Module SimDag do SimGrid**. Disponível em: <http://simgrid.gforge.inria.fr/doc/group__SD__API.html>. Acesso em 19 de setembro de 2010;

[SimGrid, (1999)(1)] **SimGrid Project**. Disponível em <http://simgrid.gforge.inria.fr/doc/index.html>. Acesso em: 19 de setembro de 2010;

[SimGrid, (1999)(2)] **História do SimGrid**. Disponível em: <http://simgrid.gforge.inria.fr/doc/history.html>. Acesso em 19 de setembro de 2010;

[SMPI, (1999)] **Module SMPI do SimGrid**. Disponível em: <http://simgrid.gforge.inria.fr/doc/group__SMPI__API.html>. Acesso em 19 de setembro de 2010;

[Soares, (1992)] SOARES, L. F. G., **Modelagem e Simulação Discreta de Sistemas**, São Paulo: Campus Ltda, 1992;

[SULISTIO *et al.*, (2008)] SULISTIO, A., CIBEJ, U., VENUGOPAL, S., ROBIC, B. e BUYYA, R., **A Toolkit for Modelling and Simulating Data Grids: An Extension to GridSim**, Concurrency and Computation: Practice and Experience (CCPE), Online ISSN: 1532-0634, Printed ISSN: 1532-0626, 20(13): 1591-1609, Wiley Press, New York, USA, Sep. 2008;

[SURF, (1999)] **Module SURF do SimGrid**. Disponível em: <http://simgrid.gforge.inria.fr/doc/group__SURF__API.html>. Acesso em 19 de setembro de 2010;

[Takefusa *et al*., (1999)] TAKEFUSA, A., AIDA, K. e MATSUOKA, S., **Overview of a performance evaluation system for global computing scheduling algorithms**. In: HPDC ’99: Proceedings of the 8th IEEE International Symposium on High Performance Distributed Computing. Washington, DC, Estados Unidos: IEEE Computer Society, 1999. p. 11. ISBN 0-7695-0287-3;

Apêndice A – Exemplos de Modelos SimGrid

Nesta seção são apresentados exemplos de conversões de modelos SimGrid e modelos icônicos em formato de texto plano referentes aos exemplos descritos no capítulo 4.

Exemplo de Modelo SimGrid 1

A seguir é apresentado um exemplo de modelo SimGrid sem erros.

```
<?xml version='1.0'?>
<!DOCTYPE platform_description SYSTEM "surfxml.dtd">
<platform_description version="1">
  <!-- The master process (with some arguments) -->
  <process host="gspd-fe" function="master">
    <argument value="20"/>       <!-- Number of tasks -->
```

```
    <argument value="20000"/>    <!-- Max computation size of tasks -->
    <argument value="20"/>       <!-- Max communication size of tasks -->
    <argument value="gspd-node1"/>   <!--First cluster -->
    <argument value="gspd-node2"/>
    <argument value="gspd-node3"/>
    <argument value="gspd-node4"/>
    <argument value="gspd-node5"/>
    <argument value="gspd-node6"/>
    <argument value="gspd-node7"/>
    <argument value="gspd-node8"/>
  </process>

  <process host="gspd-node1" function="slave"/>
  <process host="gspd-node2" function="slave"/>
  <process host="gspd-node3" function="slave"/>
  <process host="gspd-node4" function="slave"/>
  <process host="gspd-node5" function="slave"/>
  <process host="gspd-node6" function="slave"/>
  <process host="gspd-node7" function="slave"/>
  <process host="gspd-node8" function="slave"/>
</platform_description>

<?xml version='1.0'?>
<!DOCTYPE platform_description SYSTEM "surfxml.dtd">
<platform_description version="1">
```

```
<!-- poder - Mflop/s   banda - Mb/s  latencia s -->
<cpu name="gspd-fe" power="10000"/>
<cpu name="gspd-node1" power="10000"/>
<cpu name="gspd-node2" power="10000"/>
<cpu name="gspd-node3" power="10000"/>
<cpu name="gspd-node4" power="10000"/>
<cpu name="gspd-node5" power="10000"/>
<cpu name="gspd-node6" power="10000"/>
<cpu name="gspd-node7" power="10000"/>
<cpu name="gspd-node8" power="10000"/>

<!-- conexoes entre os escalonador e clusters de  100Mbps e 1Gbps -->
<network_link name="lan" bandwidth="100" latency="0.001"/>

<!-- roteamento-->
<route src="gspd-fe" dst="gspd-node1"><route_element name="lan"/></route>
<route src="gspd-node1" dst="gspd-fe"><route_element name="lan"/></route>
<route src="gspd-fe" dst="gspd-node2"><route_element name="lan"/></route>
<route src="gspd-node2" dst="gspd-fe"><route_element name="lan"/></route>
<route src="gspd-fe" dst="gspd-node3"><route_element name="lan"/></route>
<route src="gspd-node3" dst="gspd-fe"><route_element name="lan"/></route>
<route src="gspd-fe" dst="gspd-node4"><route_element name="lan"/></route>
<route src="gspd-node4" dst="gspd-fe"><route_element name="lan"/></route>
<route src="gspd-fe" dst="gspd-node5"><route_element name="lan"/></route>
<route src="gspd-node5" dst="gspd-fe"><route_element name="lan"/></route>
<route src="gspd-fe" dst="gspd-node6"><route_element name="lan"/></route>
```

<route src="gspd-node6" dst="gspd-fe"><route_element name="lan"/></route>

<route src="gspd-fe" dst="gspd-node7"><route_element name="lan"/></route>

<route src="gspd-node7" dst="gspd-fe"><route_element name="lan"/></route>

<route src="gspd-fe" dst="gspd-node8"><route_element name="lan"/></route>

<route src="gspd-node8" dst="gspd-fe"><route_element name="lan"/></route>

<route src="gspd-node1" dst="gspd-node2"><route_element name="lan"/></route>

<route src="gspd-node2" dst="gspd-node1"><route_element name="lan"/></route>

<route src="gspd-node1" dst="gspd-node3"><route_element name="lan"/></route>

<route src="gspd-node3" dst="gspd-node1"><route_element name="lan"/></route>

<route src="gspd-node1" dst="gspd-node4"><route_element name="lan"/></route>

<route src="gspd-node4" dst="gspd-node1"><route_element name="lan"/></route>

<route src="gspd-node1" dst="gspd-node5"><route_element name="lan"/></route>

<route src="gspd-node5" dst="gspd-node1"><route_element name="lan"/></route>

<route src="gspd-node1" dst="gspd-node6"><route_element name="lan"/></route>

<route src="gspd-node6" dst="gspd-node1"><route_element name="lan"/></route>

<route src="gspd-node1" dst="gspd-node7"><route_element name="lan"/></route>

<route src="gspd-node7" dst="gspd-node1"><route_element name="lan"/></route>

<route src="gspd-node1" dst="gspd-node8"><route_element name="lan"/></route>

<route src="gspd-node8" dst="gspd-node1"><route_element name="lan"/></route>

<route src="gspd-node2" dst="gspd-node3"><route_element name="lan"/></route>

<route src="gspd-node3" dst="gspd-node2"><route_element name="lan"/></route>

<route src="gspd-node2" dst="gspd-node4"><route_element name="lan"/></route>

<route src="gspd-node4" dst="gspd-node2"><route_element name="lan"/></route>

<route src="gspd-node2" dst="gspd-node5"><route_element name="lan"/></route>

<route src="gspd-node5" dst="gspd-node2"><route_element name="lan"/></route>

<route src="gspd-node2" dst="gspd-node6"><route_element name="lan"/></route>

<route src="gspd-node6" dst="gspd-node2"><route_element name="lan"/></route>

<route src="gspd-node2" dst="gspd-node7"><route_element name="lan"/></route>

<route src="gspd-node7" dst="gspd-node2"><route_element name="lan"/></route>

<route src="gspd-node2" dst="gspd-node8"><route_element name="lan"/></route>

<route src="gspd-node8" dst="gspd-node2"><route_element name="lan"/></route>

<route src="gspd-node3" dst="gspd-node4"><route_element name="lan"/></route>

<route src="gspd-node4" dst="gspd-node3"><route_element name="lan"/></route>

<route src="gspd-node3" dst="gspd-node5"><route_element name="lan"/></route>

<route src="gspd-node5" dst="gspd-node3"><route_element name="lan"/></route>

<route src="gspd-node3" dst="gspd-node6"><route_element name="lan"/></route>

<route src="gspd-node6" dst="gspd-node3"><route_element name="lan"/></route>

<route src="gspd-node3" dst="gspd-node7"><route_element name="lan"/></route>

<route src="gspd-node7" dst="gspd-node3"><route_element name="lan"/></route>

<route src="gspd-node3" dst="gspd-node8"><route_element name="lan"/></route>

<route src="gspd-node8" dst="gspd-node3"><route_element name="lan"/></route>

<route src="gspd-node4" dst="gspd-node5"><route_element name="lan"/></route>

<route src="gspd-node5" dst="gspd-node4"><route_element name="lan"/></route>

<route src="gspd-node4" dst="gspd-node6"><route_element name="lan"/></route>

<route src="gspd-node6" dst="gspd-node4"><route_element name="lan"/></route>

<route src="gspd-node4" dst="gspd-node7"><route_element name="lan"/></route>

<route src="gspd-node7" dst="gspd-node4"><route_element name="lan"/></route>

<route src="gspd-node4" dst="gspd-node8"><route_element name="lan"/></route>

<route src="gspd-node8" dst="gspd-node4"><route_element name="lan"/></route>

<route src="gspd-node5" dst="gspd-node6"><route_element name="lan"/></route>

<route src="gspd-node6" dst="gspd-node5"><route_element name="lan"/></route>

<route src="gspd-node5" dst="gspd-node7"><route_element name="lan"/></route>

```
  <route src="gspd-node7" dst="gspd-node5"><route_element name="lan"/></route>
  <route src="gspd-node5" dst="gspd-node8"><route_element name="lan"/></route>
  <route src="gspd-node8" dst="gspd-node5"><route_element name="lan"/></route>

  <route src="gspd-node6" dst="gspd-node7"><route_element name="lan"/></route>
  <route src="gspd-node7" dst="gspd-node6"><route_element name="lan"/></route>
  <route src="gspd-node6" dst="gspd-node8"><route_element name="lan"/></route>
  <route src="gspd-node8" dst="gspd-node6"><route_element name="lan"/></route>
  <route src="gspd-node7" dst="gspd-node8"><route_element name="lan"/></route>
  <route src="gspd-node8" dst="gspd-node7"><route_element name="lan"/></route>
</platform_description>
```

A seguir é apresentado o modelo gerado em formato de texto plano a partir do modelo icônico da Figura 4.1, que representa o modelo SimGrid descrito acima.

```
MAQ gspd-fe 10000.0 0.0 MESTRE RoundRobin LMAQ gspd-node1 gspd-node2 gspd-node3 gspd-node4 gspd-node5 gspd-node6 gspd-node7 gspd-node8
MAQ gspd-node1 10000.0 0.0 ESCRAVO
MAQ gspd-node2 10000.0 0.0 ESCRAVO
MAQ gspd-node3 10000.0 0.0 ESCRAVO
MAQ gspd-node4 10000.0 0.0 ESCRAVO
MAQ gspd-node5 10000.0 0.0 ESCRAVO
MAQ gspd-node6 10000.0 0.0 ESCRAVO
MAQ gspd-node7 10000.0 0.0 ESCRAVO
MAQ gspd-node8 10000.0 0.0 ESCRAVO
REDE lan[1] 100.0 0.001 0.0 CONECTA gspd-fe gspd-node1
REDE lan[2] 100.0 0.001 0.0 CONECTA gspd-node1 gspd-fe
```

REDE lan[3] 100.0 0.001 0.0 CONECTA gspd-fe gspd-node2

REDE lan[4] 100.0 0.001 0.0 CONECTA gspd-node2 gspd-fe

REDE lan[5] 100.0 0.001 0.0 CONECTA gspd-fe gspd-node3

REDE lan[6] 100.0 0.001 0.0 CONECTA gspd-node3 gspd-fe

REDE lan[7] 100.0 0.001 0.0 CONECTA gspd-fe gspd-node4

REDE lan[8] 100.0 0.001 0.0 CONECTA gspd-node4 gspd-fe

REDE lan[9] 100.0 0.001 0.0 CONECTA gspd-fe gspd-node5

REDE lan[10] 100.0 0.001 0.0 CONECTA gspd-node5 gspd-fe

REDE lan[11] 100.0 0.001 0.0 CONECTA gspd-fe gspd-node6

REDE lan[12] 100.0 0.001 0.0 CONECTA gspd-node6 gspd-fe

REDE lan[13] 100.0 0.001 0.0 CONECTA gspd-fe gspd-node7

REDE lan[14] 100.0 0.001 0.0 CONECTA gspd-node7 gspd-fe

REDE lan[15] 100.0 0.001 0.0 CONECTA gspd-fe gspd-node8

REDE lan[16] 100.0 0.001 0.0 CONECTA gspd-node8 gspd-fe

REDE lan[17] 100.0 0.001 0.0 CONECTA gspd-node1 gspd-node2

REDE lan[18] 100.0 0.001 0.0 CONECTA gspd-node2 gspd-node1

REDE lan[19] 100.0 0.001 0.0 CONECTA gspd-node1 gspd-node3

REDE lan[20] 100.0 0.001 0.0 CONECTA gspd-node3 gspd-node1

REDE lan[21] 100.0 0.001 0.0 CONECTA gspd-node1 gspd-node4

REDE lan[22] 100.0 0.001 0.0 CONECTA gspd-node4 gspd-node1

REDE lan[23] 100.0 0.001 0.0 CONECTA gspd-node1 gspd-node5

REDE lan[24] 100.0 0.001 0.0 CONECTA gspd-node5 gspd-node1

REDE lan[25] 100.0 0.001 0.0 CONECTA gspd-node1 gspd-node6

REDE lan[26] 100.0 0.001 0.0 CONECTA gspd-node6 gspd-node1

REDE lan[27] 100.0 0.001 0.0 CONECTA gspd-node1 gspd-node7

REDE lan[28] 100.0 0.001 0.0 CONECTA gspd-node7 gspd-node1

REDE lan[29] 100.0 0.001 0.0 CONECTA gspd-node1 gspd-node8

REDE lan[30] 100.0 0.001 0.0 CONECTA gspd-node8 gspd-node1

REDE lan[31] 100.0 0.001 0.0 CONECTA gspd-node2 gspd-node3

REDE lan[32] 100.0 0.001 0.0 CONECTA gspd-node3 gspd-node2

REDE lan[33] 100.0 0.001 0.0 CONECTA gspd-node2 gspd-node4

REDE lan[34] 100.0 0.001 0.0 CONECTA gspd-node4 gspd-node2

REDE lan[35] 100.0 0.001 0.0 CONECTA gspd-node2 gspd-node5

REDE lan[36] 100.0 0.001 0.0 CONECTA gspd-node5 gspd-node2

REDE lan[37] 100.0 0.001 0.0 CONECTA gspd-node2 gspd-node6

REDE lan[38] 100.0 0.001 0.0 CONECTA gspd-node6 gspd-node2

REDE lan[39] 100.0 0.001 0.0 CONECTA gspd-node2 gspd-node7

REDE lan[40] 100.0 0.001 0.0 CONECTA gspd-node7 gspd-node2

REDE lan[41] 100.0 0.001 0.0 CONECTA gspd-node2 gspd-node8

REDE lan[42] 100.0 0.001 0.0 CONECTA gspd-node8 gspd-node2

REDE lan[43] 100.0 0.001 0.0 CONECTA gspd-node3 gspd-node4

REDE lan[44] 100.0 0.001 0.0 CONECTA gspd-node4 gspd-node3

REDE lan[45] 100.0 0.001 0.0 CONECTA gspd-node3 gspd-node5

REDE lan[46] 100.0 0.001 0.0 CONECTA gspd-node5 gspd-node3

REDE lan[47] 100.0 0.001 0.0 CONECTA gspd-node3 gspd-node6

REDE lan[48] 100.0 0.001 0.0 CONECTA gspd-node6 gspd-node3

REDE lan[49] 100.0 0.001 0.0 CONECTA gspd-node3 gspd-node7

REDE lan[50] 100.0 0.001 0.0 CONECTA gspd-node7 gspd-node3

REDE lan[51] 100.0 0.001 0.0 CONECTA gspd-node3 gspd-node8

REDE lan[52] 100.0 0.001 0.0 CONECTA gspd-node8 gspd-node3

REDE lan[53] 100.0 0.001 0.0 CONECTA gspd-node4 gspd-node5

REDE lan[54] 100.0 0.001 0.0 CONECTA gspd-node5 gspd-node4

```
REDE lan[55] 100.0 0.001 0.0 CONECTA gspd-node4 gspd-node6
REDE lan[56] 100.0 0.001 0.0 CONECTA gspd-node6 gspd-node4
REDE lan[57] 100.0 0.001 0.0 CONECTA gspd-node4 gspd-node7
REDE lan[58] 100.0 0.001 0.0 CONECTA gspd-node7 gspd-node4
REDE lan[59] 100.0 0.001 0.0 CONECTA gspd-node4 gspd-node8
REDE lan[60] 100.0 0.001 0.0 CONECTA gspd-node8 gspd-node4
REDE lan[61] 100.0 0.001 0.0 CONECTA gspd-node5 gspd-node6
REDE lan[62] 100.0 0.001 0.0 CONECTA gspd-node6 gspd-node5
REDE lan[63] 100.0 0.001 0.0 CONECTA gspd-node5 gspd-node7
REDE lan[64] 100.0 0.001 0.0 CONECTA gspd-node7 gspd-node5
REDE lan[65] 100.0 0.001 0.0 CONECTA gspd-node5 gspd-node8
REDE lan[66] 100.0 0.001 0.0 CONECTA gspd-node8 gspd-node5
REDE lan[67] 100.0 0.001 0.0 CONECTA gspd-node6 gspd-node7
REDE lan[68] 100.0 0.001 0.0 CONECTA gspd-node7 gspd-node6
REDE lan[69] 100.0 0.001 0.0 CONECTA gspd-node6 gspd-node8
REDE lan[70] 100.0 0.001 0.0 CONECTA gspd-node8 gspd-node6
REDE lan[71] 100.0 0.001 0.0 CONECTA gspd-node7 gspd-node8
REDE lan[72] 100.0 0.001 0.0 CONECTA gspd-node8 gspd-node7
CARGA MAQUINA gspd-fe 20 20000.0 0.0 20.0 0.0 POISSON
```

Exemplo de Modelo SimGrid 2

A seguir é apresentado outro exemplo de modelo SimGrid sem erros.

```
<?xml version='1.0'?>
<!DOCTYPE platform_description SYSTEM "surfxml.dtd">
<platform_description version="1">
```

```
<!-- The master process (with some arguments) -->
<process host="gspd-fe" function="master">
  <argument value="10"/>       <!-- Number of tasks -->
  <argument value="15000"/>    <!-- Max computation size of tasks -->
  <argument value="50"/>       <!-- Max communication size of tasks -->
  <argument value="gspd-node1"/>   <!--First cluster -->
  <argument value="gspd-node2"/>
  <argument value="gspd-node3"/>
  <argument value="gspd-node4"/>
  <argument value="gspd-node5"/>
  <argument value="gspd-node6"/>
  <argument value="gspd-node7"/>
  <argument value="gspd-node8"/>
        <argument value="gspd-node9"/>
        <argument value="gspd-node10"/>
  <argument value="gspd-node11"/>   <!--First cluster -->
  <argument value="gspd-node12"/>
  <argument value="gspd-node13"/>
  <argument value="gspd-node14"/>
  <argument value="gspd-node15"/>
  <argument value="gspd-node16"/>
  <argument value="gspd-node17"/>
  <argument value="gspd-node18"/>

</process>
```

```
  <process host="gspd-node1" function="slave"/>
  <process host="gspd-node2" function="slave"/>
  <process host="gspd-node3" function="slave"/>
  <process host="gspd-node4" function="slave"/>
  <process host="gspd-node5" function="slave"/>
  <process host="gspd-node6" function="slave"/>
  <process host="gspd-node7" function="slave"/>
  <process host="gspd-node8" function="slave"/>
  <process host="gspd-node9" function="slave"/>
  <process host="gspd-node10" function="slave"/>
  <process host="gspd-node11" function="slave"/>
  <process host="gspd-node12" function="slave"/>
  <process host="gspd-node13" function="slave"/>
  <process host="gspd-node14" function="slave"/>
  <process host="gspd-node15" function="slave"/>
  <process host="gspd-node16" function="slave"/>
  <process host="gspd-node17" function="slave"/>
  <process host="gspd-node18" function="slave"/>

</platform_description>

<?xml version='1.0'?>
<!DOCTYPE platform_description SYSTEM "surfxml.dtd">
<platform_description version="1">

  <!-- poder - Mflop/s   banda - Mb/s  latencia s -->
```

```
<cpu name="gspd-fe" power="20000"/>
<cpu name="gspd-node1" power="20000"/>
<cpu name="gspd-node2" power="20000"/>
<cpu name="gspd-node3" power="20000"/>
<cpu name="gspd-node4" power="20000"/>
<cpu name="gspd-node5" power="20000"/>
<cpu name="gspd-node6" power="20000"/>
<cpu name="gspd-node7" power="20000"/>
<cpu name="gspd-node8" power="20000"/>
<cpu name="gspd-node9" power="20000"/>
<cpu name="gspd-node10" power="20000"/>
<cpu name="gspd-node11" power="20000"/>
<cpu name="gspd-node12" power="20000"/>
<cpu name="gspd-node13" power="20000"/>
<cpu name="gspd-node14" power="20000"/>
<cpu name="gspd-node15" power="20000"/>
<cpu name="gspd-node16" power="20000"/>
<cpu name="gspd-node17" power="20000"/>
<cpu name="gspd-node18" power="20000"/>

<!-- conexoes entre os escalonador e clusters de  100Mbps e 1Gbps -->
<network_link name="lan" bandwidth="200" latency="0.0001"/>

<!-- roteamento-->
<route src="gspd-fe" dst="gspd-node1"><route_element name="lan"/></route>
<route src="gspd-node1" dst="gspd-fe"><route_element name="lan"/></route>
```

<route src="gspd-fe" dst="gspd-node2"><route_element name="lan"/></route>

<route src="gspd-node2" dst="gspd-fe"><route_element name="lan"/></route>

<route src="gspd-fe" dst="gspd-node3"><route_element name="lan"/></route>

<route src="gspd-node3" dst="gspd-fe"><route_element name="lan"/></route>

<route src="gspd-fe" dst="gspd-node4"><route_element name="lan"/></route>

<route src="gspd-node4" dst="gspd-fe"><route_element name="lan"/></route>

<route src="gspd-fe" dst="gspd-node5"><route_element name="lan"/></route>

<route src="gspd-node5" dst="gspd-fe"><route_element name="lan"/></route>

<route src="gspd-fe" dst="gspd-node6"><route_element name="lan"/></route>

<route src="gspd-node6" dst="gspd-fe"><route_element name="lan"/></route>

<route src="gspd-fe" dst="gspd-node7"><route_element name="lan"/></route>

<route src="gspd-node7" dst="gspd-fe"><route_element name="lan"/></route>

<route src="gspd-fe" dst="gspd-node8"><route_element name="lan"/></route>

<route src="gspd-node8" dst="gspd-fe"><route_element name="lan"/></route>

<route src="gspd-fe" dst="gspd-node9"><route_element name="lan"/></route>

<route src="gspd-node9" dst="gspd-fe"><route_element name="lan"/></route>

<route src="gspd-fe" dst="gspd-node10"><route_element name="lan"/></route>

<route src="gspd-node10" dst="gspd-fe"><route_element name="lan"/></route>

<route src="gspd-fe" dst="gspd-node11"><route_element name="lan"/></route>

<route src="gspd-node11" dst="gspd-fe"><route_element name="lan"/></route>

<route src="gspd-fe" dst="gspd-node12"><route_element name="lan"/></route>

<route src="gspd-node12" dst="gspd-fe"><route_element name="lan"/></route>

<route src="gspd-fe" dst="gspd-node13"><route_element name="lan"/></route>

<route src="gspd-node13" dst="gspd-fe"><route_element name="lan"/></route>

<route src="gspd-fe" dst="gspd-node14"><route_element name="lan"/></route>

<route src="gspd-node14" dst="gspd-fe"><route_element name="lan"/></route>

<route src="gspd-fe" dst="gspd-node15"><route_element name="lan"/></route>

<route src="gspd-node15" dst="gspd-fe"><route_element name="lan"/></route>

<route src="gspd-fe" dst="gspd-node16"><route_element name="lan"/></route>

<route src="gspd-node16" dst="gspd-fe"><route_element name="lan"/></route>

<route src="gspd-fe" dst="gspd-node17"><route_element name="lan"/></route>

<route src="gspd-node17" dst="gspd-fe"><route_element name="lan"/></route>

<route src="gspd-fe" dst="gspd-node18"><route_element name="lan"/></route>

<route src="gspd-node18" dst="gspd-fe"><route_element name="lan"/></route>

<route src="gspd-node1" dst="gspd-node2"><route_element name="lan"/></route>

<route src="gspd-node2" dst="gspd-node1"><route_element name="lan"/></route>

<route src="gspd-node1" dst="gspd-node3"><route_element name="lan"/></route>

<route src="gspd-node3" dst="gspd-node1"><route_element name="lan"/></route>

<route src="gspd-node1" dst="gspd-node4"><route_element name="lan"/></route>

<route src="gspd-node4" dst="gspd-node1"><route_element name="lan"/></route>

<route src="gspd-node1" dst="gspd-node5"><route_element name="lan"/></route>

<route src="gspd-node5" dst="gspd-node1"><route_element name="lan"/></route>

<route src="gspd-node1" dst="gspd-node6"><route_element name="lan"/></route>

<route src="gspd-node6" dst="gspd-node1"><route_element name="lan"/></route>

<route src="gspd-node1" dst="gspd-node7"><route_element name="lan"/></route>

<route src="gspd-node7" dst="gspd-node1"><route_element name="lan"/></route>

<route src="gspd-node1" dst="gspd-node8"><route_element name="lan"/></route>

<route src="gspd-node8" dst="gspd-node1"><route_element name="lan"/></route>

<route src="gspd-node2" dst="gspd-node3"><route_element name="lan"/></route>

<route src="gspd-node3" dst="gspd-node2"><route_element name="lan"/></route>

<route src="gspd-node2" dst="gspd-node4"><route_element name="lan"/></route>

<route src="gspd-node4" dst="gspd-node2"><route_element name="lan"/></route>

<route src="gspd-node2" dst="gspd-node5"><route_element name="lan"/></route>

<route src="gspd-node5" dst="gspd-node2"><route_element name="lan"/></route>

<route src="gspd-node2" dst="gspd-node6"><route_element name="lan"/></route>

<route src="gspd-node6" dst="gspd-node2"><route_element name="lan"/></route>

<route src="gspd-node2" dst="gspd-node7"><route_element name="lan"/></route>

<route src="gspd-node7" dst="gspd-node2"><route_element name="lan"/></route>

<route src="gspd-node2" dst="gspd-node8"><route_element name="lan"/></route>

<route src="gspd-node8" dst="gspd-node2"><route_element name="lan"/></route>

<route src="gspd-node3" dst="gspd-node4"><route_element name="lan"/></route>

<route src="gspd-node4" dst="gspd-node3"><route_element name="lan"/></route>

<route src="gspd-node3" dst="gspd-node5"><route_element name="lan"/></route>

<route src="gspd-node5" dst="gspd-node3"><route_element name="lan"/></route>

<route src="gspd-node3" dst="gspd-node6"><route_element name="lan"/></route>

<route src="gspd-node6" dst="gspd-node3"><route_element name="lan"/></route>

<route src="gspd-node3" dst="gspd-node7"><route_element name="lan"/></route>

<route src="gspd-node7" dst="gspd-node3"><route_element name="lan"/></route>

<route src="gspd-node3" dst="gspd-node8"><route_element name="lan"/></route>

<route src="gspd-node8" dst="gspd-node3"><route_element name="lan"/></route>

<route src="gspd-node4" dst="gspd-node5"><route_element name="lan"/></route>

<route src="gspd-node5" dst="gspd-node4"><route_element name="lan"/></route>

<route src="gspd-node4" dst="gspd-node6"><route_element name="lan"/></route>

```
 <route src="gspd-node6" dst="gspd-node4"><route_element name="lan"/></route>
 <route src="gspd-node4" dst="gspd-node7"><route_element name="lan"/></route>
 <route src="gspd-node7" dst="gspd-node4"><route_element name="lan"/></route>
 <route src="gspd-node4" dst="gspd-node8"><route_element name="lan"/></route>
 <route src="gspd-node8" dst="gspd-node4"><route_element name="lan"/></route>

 <route src="gspd-node5" dst="gspd-node6"><route_element name="lan"/></route>
 <route src="gspd-node6" dst="gspd-node5"><route_element name="lan"/></route>
 <route src="gspd-node5" dst="gspd-node7"><route_element name="lan"/></route>
 <route src="gspd-node7" dst="gspd-node5"><route_element name="lan"/></route>
 <route src="gspd-node5" dst="gspd-node8"><route_element name="lan"/></route>
 <route src="gspd-node8" dst="gspd-node5"><route_element name="lan"/></route>

 <route src="gspd-node6" dst="gspd-node7"><route_element name="lan"/></route>
 <route src="gspd-node7" dst="gspd-node6"><route_element name="lan"/></route>
 <route src="gspd-node6" dst="gspd-node8"><route_element name="lan"/></route>
 <route src="gspd-node8" dst="gspd-node6"><route_element name="lan"/></route>

 <route src="gspd-node7" dst="gspd-node8"><route_element name="lan"/></route>
 <route src="gspd-node8" dst="gspd-node7"><route_element name="lan"/></route>

</platform_description>
```

A seguir é apresentado o modelo gerado em formato de texto plano a partir do modelo icônico da Figura 4.2, que representa o modelo SimGrid descrito acima.

MAQ gspd-fe 20000.0 0.0 MESTRE RoundRobin LMAQ gspd-node1 gspd-node2 gspd-node3 gspd-node4 gspd-node5 gspd-node6 gspd-node7 gspd-node8 gspd-node9 gspd-node10 gspd-node11 gspd-node12 gspd-node13 gspd-node14 gspd-node15 gspd-node16 gspd-node17 gspd-node18

MAQ gspd-node1 20000.0 0.0 ESCRAVO

MAQ gspd-node2 20000.0 0.0 ESCRAVO

MAQ gspd-node3 20000.0 0.0 ESCRAVO

MAQ gspd-node4 20000.0 0.0 ESCRAVO

MAQ gspd-node5 20000.0 0.0 ESCRAVO

MAQ gspd-node6 20000.0 0.0 ESCRAVO

MAQ gspd-node7 20000.0 0.0 ESCRAVO

MAQ gspd-node8 20000.0 0.0 ESCRAVO

MAQ gspd-node9 20000.0 0.0 ESCRAVO

MAQ gspd-node10 20000.0 0.0 ESCRAVO

MAQ gspd-node11 20000.0 0.0 ESCRAVO

MAQ gspd-node12 20000.0 0.0 ESCRAVO

MAQ gspd-node13 20000.0 0.0 ESCRAVO

MAQ gspd-node14 20000.0 0.0 ESCRAVO

MAQ gspd-node15 20000.0 0.0 ESCRAVO

MAQ gspd-node16 20000.0 0.0 ESCRAVO

MAQ gspd-node17 20000.0 0.0 ESCRAVO

MAQ gspd-node18 20000.0 0.0 ESCRAVO

REDE lan[1] 200.0 0.0001 0.0 CONECTA gspd-fe gspd-node1

REDE lan[2] 200.0 0.0001 0.0 CONECTA gspd-node1 gspd-fe

REDE lan[3] 200.0 0.0001 0.0 CONECTA gspd-fe gspd-node2

REDE lan[4] 200.0 0.0001 0.0 CONECTA gspd-node2 gspd-fe

REDE lan[5] 200.0 0.0001 0.0 CONECTA gspd-fe gspd-node3

REDE lan[6] 200.0 0.0001 0.0 CONECTA gspd-node3 gspd-fe

REDE lan[7] 200.0 0.0001 0.0 CONECTA gspd-fe gspd-node4

REDE lan[8] 200.0 0.0001 0.0 CONECTA gspd-node4 gspd-fe

REDE lan[9] 200.0 0.0001 0.0 CONECTA gspd-fe gspd-node5

REDE lan[10] 200.0 0.0001 0.0 CONECTA gspd-node5 gspd-fe

REDE lan[11] 200.0 0.0001 0.0 CONECTA gspd-fe gspd-node6

REDE lan[12] 200.0 0.0001 0.0 CONECTA gspd-node6 gspd-fe

REDE lan[13] 200.0 0.0001 0.0 CONECTA gspd-fe gspd-node7

REDE lan[14] 200.0 0.0001 0.0 CONECTA gspd-node7 gspd-fe

REDE lan[15] 200.0 0.0001 0.0 CONECTA gspd-fe gspd-node8

REDE lan[16] 200.0 0.0001 0.0 CONECTA gspd-node8 gspd-fe

REDE lan[17] 200.0 0.0001 0.0 CONECTA gspd-fe gspd-node9

REDE lan[18] 200.0 0.0001 0.0 CONECTA gspd-node9 gspd-fe

REDE lan[19] 200.0 0.0001 0.0 CONECTA gspd-fe gspd-node10

REDE lan[20] 200.0 0.0001 0.0 CONECTA gspd-node10 gspd-fe

REDE lan[21] 200.0 0.0001 0.0 CONECTA gspd-fe gspd-node11

REDE lan[22] 200.0 0.0001 0.0 CONECTA gspd-node11 gspd-fe

REDE lan[23] 200.0 0.0001 0.0 CONECTA gspd-fe gspd-node12

REDE lan[24] 200.0 0.0001 0.0 CONECTA gspd-node12 gspd-fe

REDE lan[25] 200.0 0.0001 0.0 CONECTA gspd-fe gspd-node13

REDE lan[26] 200.0 0.0001 0.0 CONECTA gspd-node13 gspd-fe

REDE lan[27] 200.0 0.0001 0.0 CONECTA gspd-fe gspd-node14

REDE lan[28] 200.0 0.0001 0.0 CONECTA gspd-node14 gspd-fe

REDE lan[29] 200.0 0.0001 0.0 CONECTA gspd-fe gspd-node15

REDE lan[30] 200.0 0.0001 0.0 CONECTA gspd-node15 gspd-fe

REDE lan[31] 200.0 0.0001 0.0 CONECTA gspd-fe gspd-node16

REDE lan[32] 200.0 0.0001 0.0 CONECTA gspd-node16 gspd-fe

REDE lan[33] 200.0 0.0001 0.0 CONECTA gspd-fe gspd-node17

REDE lan[34] 200.0 0.0001 0.0 CONECTA gspd-node17 gspd-fe

REDE lan[35] 200.0 0.0001 0.0 CONECTA gspd-fe gspd-node18

REDE lan[36] 200.0 0.0001 0.0 CONECTA gspd-node18 gspd-fe

REDE lan[37] 200.0 0.0001 0.0 CONECTA gspd-node1 gspd-node2

REDE lan[38] 200.0 0.0001 0.0 CONECTA gspd-node2 gspd-node1

REDE lan[39] 200.0 0.0001 0.0 CONECTA gspd-node1 gspd-node3

REDE lan[40] 200.0 0.0001 0.0 CONECTA gspd-node3 gspd-node1

REDE lan[41] 200.0 0.0001 0.0 CONECTA gspd-node1 gspd-node4

REDE lan[42] 200.0 0.0001 0.0 CONECTA gspd-node4 gspd-node1

REDE lan[43] 200.0 0.0001 0.0 CONECTA gspd-node1 gspd-node5

REDE lan[44] 200.0 0.0001 0.0 CONECTA gspd-node5 gspd-node1

REDE lan[45] 200.0 0.0001 0.0 CONECTA gspd-node1 gspd-node6

REDE lan[46] 200.0 0.0001 0.0 CONECTA gspd-node6 gspd-node1

REDE lan[47] 200.0 0.0001 0.0 CONECTA gspd-node1 gspd-node7

REDE lan[48] 200.0 0.0001 0.0 CONECTA gspd-node7 gspd-node1

REDE lan[49] 200.0 0.0001 0.0 CONECTA gspd-node1 gspd-node8

REDE lan[50] 200.0 0.0001 0.0 CONECTA gspd-node8 gspd-node1

REDE lan[51] 200.0 0.0001 0.0 CONECTA gspd-node2 gspd-node3

REDE lan[52] 200.0 0.0001 0.0 CONECTA gspd-node3 gspd-node2

REDE lan[53] 200.0 0.0001 0.0 CONECTA gspd-node2 gspd-node4

REDE lan[54] 200.0 0.0001 0.0 CONECTA gspd-node4 gspd-node2

REDE lan[55] 200.0 0.0001 0.0 CONECTA gspd-node2 gspd-node5

REDE lan[56] 200.0 0.0001 0.0 CONECTA gspd-node5 gspd-node2

REDE lan[57] 200.0 0.0001 0.0 CONECTA gspd-node2 gspd-node6

REDE lan[58] 200.0 0.0001 0.0 CONECTA gspd-node6 gspd-node2

REDE lan[59] 200.0 0.0001 0.0 CONECTA gspd-node2 gspd-node7

REDE lan[60] 200.0 0.0001 0.0 CONECTA gspd-node7 gspd-node2

REDE lan[61] 200.0 0.0001 0.0 CONECTA gspd-node2 gspd-node8

REDE lan[62] 200.0 0.0001 0.0 CONECTA gspd-node8 gspd-node2

REDE lan[63] 200.0 0.0001 0.0 CONECTA gspd-node3 gspd-node4

REDE lan[64] 200.0 0.0001 0.0 CONECTA gspd-node4 gspd-node3

REDE lan[65] 200.0 0.0001 0.0 CONECTA gspd-node3 gspd-node5

REDE lan[66] 200.0 0.0001 0.0 CONECTA gspd-node5 gspd-node3

REDE lan[67] 200.0 0.0001 0.0 CONECTA gspd-node3 gspd-node6

REDE lan[68] 200.0 0.0001 0.0 CONECTA gspd-node6 gspd-node3

REDE lan[69] 200.0 0.0001 0.0 CONECTA gspd-node3 gspd-node7

REDE lan[70] 200.0 0.0001 0.0 CONECTA gspd-node7 gspd-node3

REDE lan[71] 200.0 0.0001 0.0 CONECTA gspd-node3 gspd-node8

REDE lan[72] 200.0 0.0001 0.0 CONECTA gspd-node8 gspd-node3

REDE lan[73] 200.0 0.0001 0.0 CONECTA gspd-node4 gspd-node5

REDE lan[74] 200.0 0.0001 0.0 CONECTA gspd-node5 gspd-node4

REDE lan[75] 200.0 0.0001 0.0 CONECTA gspd-node4 gspd-node6

REDE lan[76] 200.0 0.0001 0.0 CONECTA gspd-node6 gspd-node4

REDE lan[77] 200.0 0.0001 0.0 CONECTA gspd-node4 gspd-node7

REDE lan[78] 200.0 0.0001 0.0 CONECTA gspd-node7 gspd-node4

REDE lan[79] 200.0 0.0001 0.0 CONECTA gspd-node4 gspd-node8

REDE lan[80] 200.0 0.0001 0.0 CONECTA gspd-node8 gspd-node4

REDE lan[81] 200.0 0.0001 0.0 CONECTA gspd-node5 gspd-node6

REDE lan[82] 200.0 0.0001 0.0 CONECTA gspd-node6 gspd-node5

```
REDE lan[83] 200.0 0.0001 0.0 CONECTA gspd-node5 gspd-node7
REDE lan[84] 200.0 0.0001 0.0 CONECTA gspd-node7 gspd-node5
REDE lan[85] 200.0 0.0001 0.0 CONECTA gspd-node5 gspd-node8
REDE lan[86] 200.0 0.0001 0.0 CONECTA gspd-node8 gspd-node5
REDE lan[87] 200.0 0.0001 0.0 CONECTA gspd-node6 gspd-node7
REDE lan[88] 200.0 0.0001 0.0 CONECTA gspd-node7 gspd-node6
REDE lan[89] 200.0 0.0001 0.0 CONECTA gspd-node6 gspd-node8
REDE lan[90] 200.0 0.0001 0.0 CONECTA gspd-node8 gspd-node6
REDE lan[91] 200.0 0.0001 0.0 CONECTA gspd-node7 gspd-node8
REDE lan[92] 200.0 0.0001 0.0 CONECTA gspd-node8 gspd-node7
CARGA MAQUINA gspd-fe 10 15000.0 0.0 50.0 0.0 POISSON
```

Exemplo de Modelo SimGrid com Erros

A seguir é apresentado o exemplo de modelo SimGrid com erros léxicos, sintáticos e semânticos.

```
<?xml version='1.0'?>
<!DOCTYPE platform_description SYSTEM "surfxml.dtd">
<platform_description version="1">
  <!-- The master process (with some arguments) -->
  <process host="gspdç-fe" function="master">
    <argument value="50"/>       <!-- Number of tasks -->
    <argument value="50000"/>    <!-- Max computation size of tasks -->
    <argument value="60"/>       <!-- Max communication size of tasks -->
    <argument value="1gspd-node1"/>   <!--First cluster -->
    <argument value="gspd-node2"/>
```

```
  </process>

  <process host="1gspd-node1" function="slave"/>
  <process host="gspd-node2" function="slave"/>
  <process host="gspd-node2" function="slave"/>
  <process host="gspd-node4" function="slave"/
</platform_description>
<?xml version='1.0'?>
<!DOCTYPE platform_description SYSTEM "surfxml.dtd">
<platform_description version="1">

  <!-- poder - Mflop/s   banda - Mb/s  latencia s -->
  <cpu name="gspdç-fe" power="40000"/>
  <cpu name="1gspd-node1" power="20000"/>
  <cpu name="gspd-node2" power="30000"/>
  <cpu name="gspd-node3" power="10000"/>

  <!-- conexoes entre os escalonador e clusters de  100Mbps e 1Gbps -->
  <network_link name="lan" bandwidth="100" latency="0.001"/>

  <!-- roteamento-->
  <route src="gspdç-fe" dst="1gspd-node1"><route_element name="lan"/></route>
  <route src="1gspd-node1" dst="gspdç-fe"><route_element name="lan"/></route>
```

<route src="gspdç-fe" dst="gspd-node2"><route_element name="lan"/></route>

<route src="gspd-node2" dst="gspdç-fe"><route_element name="lan"/></route>

<route src="gspdç-fe" dst="gspd-node3"><route_element name="lan"/></route>

<route src="gspd-node3" dst="gspdç-fe"><route_element name="lan"/></route>

</platform_description>

Apêndice B – Exemplo de Modelo do Sistema

Nesta seção é apresentado um exemplo de modelo do sistema.

MAQ icon2 111.0 0.1 ESCRAVO

MAQ icon7 111.0 0.1 ESCRAVO

MAQ icon4 111.0 0.1 ESCRAVO

MAQ icon5 111.0 0.1 ESCRAVO

MAQ icon3 111.0 0.1 ESCRAVO

MAQ icon6 111.0 0.1 ESCRAVO

MAQ icon8 111.0 0.1 MESTRE RoundRobin LMAQ icon1 icon2 icon3 icon4 icon5 icon6 icon7

CLUSTER icon1 1000 1000000.0 0.1 0.1 RoundRobin

INET icon0 10000.0 0.1 0.1

REDE icon10 1000.0 0.1 0.1 CONECTA icon8 icon2

REDE lan9 1000.0 0.1 0.1 CONECTA icon2 icon8

REDE lan20 1000.0 0.1 0.1 CONECTA icon8 icon7

REDE icon21 10000.0 0.1 0.1 CONECTA icon0 icon8

REDE lan15 1000.0 0.1 0.1 CONECTA icon7 icon8

REDE icon24 10000.0 0.1 0.1 CONECTA icon1 icon0

REDE lan16 1000.0 0.1 0.1 CONECTA icon8 icon3

REDE lan14 1000.0 0.1 0.1 CONECTA icon6 icon8

REDE lan17 1000.0 0.1 0.1 CONECTA icon8 icon4

REDE lan11 1000.0 0.1 0.1 CONECTA icon3 icon8

REDE lan19 1000.0 0.1 0.1 CONECTA icon8 icon6

REDE lan18 1000.0 0.1 0.1 CONECTA icon8 icon5

REDE lan13 1000.0 0.1 0.1 CONECTA icon5 icon8

REDE lan12 1000.0 0.1 0.1 CONECTA icon4 icon8

REDE icon22 10000.0 0.1 0.1 CONECTA icon8 icon0

REDE icon23 10000.0 0.1 0.1 CONECTA icon0 icon1

CARGA RANDOM

0 10 100 1.0

0 10 100 1.0

0 10 100

Apêndice C – Exemplo de Modelo Simulável

Nesta seção é apresentado um exemplo de modelo simulável correspondente ao modelo de sistema do Apêndice B.

MODELO

TAREFA

RANDOM 0 10 100 1.0

0 10 100 1.0

0 10 100

FIM_TAREFA

CENTROS_DE_SERVICOS

CS_0 cs_icon6 1 1 FILAS fila_icon6 SERVIDORES serv_icon6 0 111.0 0.1 ESCRAVO

CS_0 cs_icon7 1 1 FILAS fila_icon7 SERVIDORES serv_icon7 0 111.0 0.1 ESCRAVO

CS_0 cs_icon4 1 1 FILAS fila_icon4 SERVIDORES serv_icon4 0 111.0 0.1 ESCRAVO

CS_2 cs_icon10 1 1 FILAS fila_icon10 SERVIDORES serv_icon10 1 1000.0 0.1 0.1

CS_2 cs_lan16 1 1 FILAS fila_lan16 SERVIDORES serv_lan16 1 1000.0 0.1 0.1

CS_0 cs_icon2 1 1 FILAS fila_icon2 SERVIDORES serv_icon2 0 111.0 0.1 ESCRAVO

CS_2 cs_icon21 1 1 FILAS fila_icon21 SERVIDORES serv_icon21 1 10000.0 0.1 0.1

CS_0 cs_icon5 1 1 FILAS fila_icon5 SERVIDORES serv_icon5 0 111.0 0.1 ESCRAVO

CS_2 cs_lan12 1 1 FILAS fila_lan12 SERVIDORES serv_lan12 1 1000.0 0.1 0.1

CS_2 cs_icon22 1 1 FILAS fila_icon22 SERVIDORES serv_icon22 1 10000.0 0.1 0.1

CS_1 cs_icon1 2 1000 RoundRobin FILAS fila_0_icon1 fila_1_icon1 SERVIDORES serv_icon1 0 1000000.0 0.1 0.1

CS_0 cs_icon8 1 1 FILAS fila_icon8 SERVIDORES serv_icon8 0 111.0 0.1 MESTRE RoundRobin LMAQ icon1 icon2 icon3 icon4 icon5 icon6 icon7

CS_3 cs_icon0 1 1 FILAS fila_icon0 SERVIDORES serv_icon0 1 10000.0 0.1 0.1

CS_2 cs_lan17 1 1 FILAS fila_lan17 SERVIDORES serv_lan17 1 1000.0 0.1 0.1

CS_2 cs_lan19 1 1 FILAS fila_lan19 SERVIDORES serv_lan19 1 1000.0 0.1 0.1

CS_2 cs_lan15 1 1 FILAS fila_lan15 SERVIDORES serv_lan15 1 1000.0 0.1 0.1

CS_2 cs_lan18 1 1 FILAS fila_lan18 SERVIDORES serv_lan18 1 1000.0 0.1 0.1

CS_2 cs_lan20 1 1 FILAS fila_lan20 SERVIDORES serv_lan20 1 1000.0 0.1 0.1

CS_2 cs_lan9 1 1 FILAS fila_lan9 SERVIDORES serv_lan9 1 1000.0 0.1 0.1

CS_0 cs_icon3 1 1 FILAS fila_icon3 SERVIDORES serv_icon3 0 111.0 0.1 ESCRAVO

CS_2 cs_lan14 1 1 FILAS fila_lan14 SERVIDORES serv_lan14 1 1000.0 0.1 0.1

CS_2 cs_lan11 1 1 FILAS fila_lan11 SERVIDORES serv_lan11 1 1000.0 0.1 0.1

CS_2 cs_icon23 1 1 FILAS fila_icon23 SERVIDORES serv_icon23 1 10000.0 0.1 0.1

CS_2 cs_icon24 1 1 FILAS fila_icon24 SERVIDORES serv_icon24 1 10000.0 0.1 0.1

CS_2 cs_lan13 1 1 FILAS fila_lan13 SERVIDORES serv_lan13 1 1000.0 0.1 0.1

FIM_CENTROS_DE_SERVICOS

CONEXOES

cs_icon8 cs_icon10

cs_icon10 cs_icon2

cs_icon8 cs_lan16

cs_lan16 cs_icon3

cs_icon0 cs_icon21

cs_icon21 cs_icon8

cs_icon4 cs_lan12

cs_lan12 cs_icon8

cs_icon8 cs_icon22

cs_icon22 cs_icon0

cs_icon8 cs_lan17

cs_lan17 cs_icon4

cs_icon8 cs_lan19

cs_lan19 cs_icon6

cs_icon7 cs_lan15

cs_lan15 cs_icon8

cs_icon8 cs_lan18

cs_lan18 cs_icon5

cs_icon8 cs_lan20

cs_lan20 cs_icon7

cs_icon2 cs_lan9

cs_lan9 cs_icon8

cs_icon6 cs_lan14

cs_lan14 cs_icon8

cs_icon3 cs_lan11

cs_lan11 cs_icon8

cs_icon0 cs_icon23

cs_icon23 cs_icon1

cs_icon1 cs_icon24

cs_icon24 cs_icon0

cs_icon5 cs_lan13

cs_lan13 cs_icon8

FIM_CONEXOES

FIM_MODELO