Modelagem conceitual de um integrador para simulação hemodinâmica em Grid

**Ramon G. Costa, Paulo G. P. Ziemer, A. T. A. Gomes,
Bruno Schulze, Pablo J. Blanco, Raúl A. Feijóo**

{ramongc, ziemer, atagomes, schulze, pjblanco, feij}@lncc.br

[1]Laboratório Nacional de Computação Científica - LNCC
Quitandinha, 25651-075 Petrópolis - RJ, Brazil

Resumo. *Avanços recentes têm mostrado que a pesquisa médica pode se beneficiar de maneira intensa através do uso de ferramentas computacionais orientadas à modelagem e simulação computacional do sistema cardiovascular humano. Isto não só atinge o entendimento da gênese e desenvolvimento de doenças cardiovasculares, mas também a avaliação de procedimentos cirúrgicos e até mesmo o treinamento de recursos humanos nas áreas médicas. Para isto é preciso superar diversas barreiras associadas à construção do modelo computacional do sistema cardiovascular. Em particular, um dos pontos de maior demanda em termos de custo computacional corresponde ao cálculo numérico aproximado do escoamento do sangue em distritos arteriais de interesse. Assim, para diminuir o tempo de computação, o software resolvedor de modelos utiliza o paradigma MPI para compartilhar a carga entre processadores diferentes. Este trabalho apresenta a modelagem conceitual do sistema integrador, que permite ao software de simulação hemodinâmica, produzir o resultado da simulação em ambiente de Grid. Este trabalho contempla o projeto arquitetural, além do projeto detalhado do sistema de software usando a UML, tais como: diagramas de classe e sequência.*

1. Introdução

Doenças relacionadas com o sistema cardiovascular humano, tais como doenças cardíacas coronarianas, são as principais causas de morte no mundo [Mackay et al. 2004]. O uso de simulação computacional pode fornecer informações exclusivas e detalhadas sobre o comportamento do fluxo de sangue para médicos e pesquisadores, dando-lhes novas perspectivas sobre os tratamentos de patologias [Blanco et al. 2009a]. O projeto HeMoLab [HeMoLab] foi criado com o objetivo de desenvolver ferramentas de simulação computacional que possam fomentar e facilitar a utilização de ferramentas de modelagem. Com isso em mente, foi desenvolvido o *software*, chamado HeMoLab, como uma extensão do *software* de visualização Kitware Paraview [Moreland 2009]. As principais funcionalidades do HeMoLab permitem a criação e personalização de modelos simplificados (modelos 1D), modelos com maior nível de detalhes (3D), e modelos acoplados (1D-3D). Os modelos produzidos pelo HeMoLab são salvos em um formato de arquivos especial e usados como entrada para o *software* que faz o cálculo da simulação, chamado SolverGP.

A experiência obtida com a simulação computacional de modelos 3D têm mostrado que o tempo de processamento exigido é da ordem de semanas ou mesmo meses

em modelos mais complexos. Desta forma, o uso de técnicas de computação de alto desempenho, como a computação em *Grid* é vital para o sucesso da adoção do *software* HeMoLab pela comunidade médica.

Neste contexto, este trabalho descreve a modelagem conceitual do sistema integrador que permite com que um sistema de simulação hemodinâmica possa ser processado em uma *Grid* Computacional. A saber, serão especificados os requisitos, a arquitetura e a correspondência entre os elementos descritos em ambas as especificações. A especificação de requisitos será expressa em linguagem natural, além do diagrama UML de caso de uso. A especificação arquitetural contempla os três principais tipos de visões arquiteturais: execução, módulo e implantação. A especificação dos módulos do subsistema integrador é expressa por meio de diagramas UML de classes e de sequência. Além dos diagramas de classe, cada classe é detalhada em linguagem natural. O trabalho contempla ainda a implementação de um protótipo do sistema integrador.

2. O projeto HeMoLab

2.1. Modelagem do Sistema Cardiovascular Humano

A simulação do sistema cardiovascular humano é realizada utilizando dois tipos de modelos: unidimensional (1D) e tridimensionais (3D). O modelo 1D representa o sistema cardiovascular de uma forma simplificada, onde as principais artérias do corpo humano são modeladas como um conjunto de linhas retas (segmentos) e pontos (terminais). Os modelos 3D são utilizados para obter um maior nível de detalhe do comportamento do fluxo de sangue dentro da artéria para uma estrutura especificada. A geometria de um modelo 3D é obtida através da análise e processamento de imagens médicas, tais como tomografia computadorizada, ressonância magnética, ultra-som intravascular, etc.

Os modelos 3D podem ser utilizados de forma independente ou integrados aos modelos 1D. Os modelos 1D-3D quando integrados são chamados de modelos acoplados [Blanco et al. 2009b]. Desta forma, os dados resultantes da simulação do modelo 1D são usados como condição de contorno para o modelo 3D. Se um modelo 3D é usado de forma autônoma, as condições de contorno devem ser configuradas em fase avançada.

A simulação computacional de um modelo é feita através da resolução (usando Método dos Elementos Finitos - FEM) de um sistema de equações de Navier Stokes que descreve o movimento dos fluidos, como: sangue dentro de uma artéria. A fim de iniciar a simulação computacional de algum modelo, o SolverGP lê o conjunto de arquivos que descrevem os parâmetros do modelo, como geometria, mecânica e parâmetros de simulação. Cada período de simulação (geralmente um período refere-se a uma batida do coração) é sub-dividido em passos de tempo (um valor típico é 640) e para cada passo de tempo, o SolverGP atualiza o arquivo que armazena os resultados da simulação (Data-out.txt) com os valores de deslocamento, velocidade e pressão em cada ponto do volume. Ele utiliza MPICH2 [Gropp et al. 2009] como a implementação MPI padrão.

A simulação computacional de modelos 1D não demanda uma potência computacional de alta performance. Esses modelos simplificados são comumente resolvido usando uma versão do SolverGP sequencial, em computadores *desktop* comuns, com tempo de resolução de aproximadamente 15 a 30 minutos. No entanto, modelos mais complexos, tais como os modelos 3D ou acoplados, podem exigir um alto poder computacional na realização das simulações. Por exemplo, o modelo de computador de uma

aorta, a maior artéria do corpo humano, necessita de aproximadamente 500.000 pontos para representar a geometria das artérias e, em seguida, dá origem a um sistema de equações com 3.500.000 incógnitas. A computação de apenas um batimento cardíaco, em tal modelo, leva aproximadamente 35 dias (com 640 passos de tempo) quando usados oito processadores em uma máquina com configuração: 2X Quad-Core Xeon 3GHz / 64GB RAM.

Após a fase de simulação computacional, técnicas de computação gráfica podem ser aplicadas aos dados produzidos, a fim de facilitar o processo de obtenção conhecimento sobre os dados. Na Figura 1, é possível ver um exemplo desse processo sobre os resultados da simulação de uma artéria que contém um aneurisma. As linhas de corrente indicam a trajetória instantânea das partículas de sangue e a magnitude da velocidade.

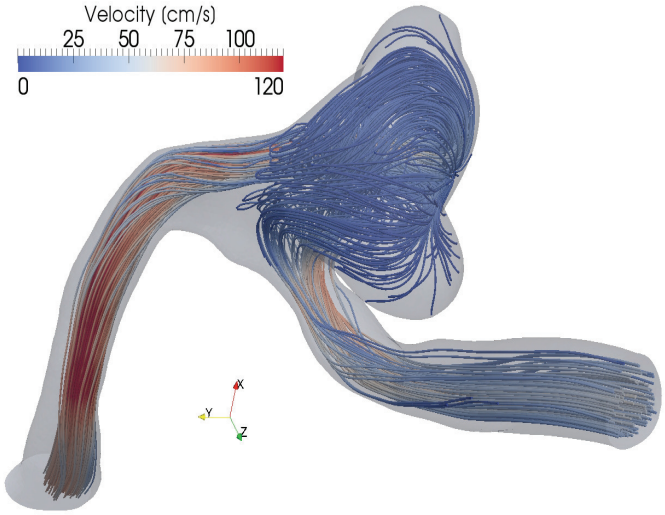

Figura 1. Exemplo de um modelo de aneurisma com resultados de simulação

3. Conceitos e Tecnologia

A fim de obter menores tempos de computação, o sistema HeMoLab usa MPI (*Message Passing Interface*) [Forum 1994] para a paralelização de simulações computacionais do sistema cardiovascular humano. Durante o processo de gridificação do HeMoLab, várias ferramentas foram usadas para apoiar a criação e configuração de serviços em *Grid*, tais como: gLite Middleware [Burke et al. 2009], Secure-Store [Scardaci and Scuderi 2007] e Watchdog [Bruno et al. 2009].

O gLite é um *middleware* para *Grids* Computacionais que fornece um *framework* para a construção de aplicações que aproveitam o poder da computação distribuída e recursos de armazenamento através da Internet.

Secure-Store [Scardaci and Scuderi 2007] é uma ferramenta usada pelo gLite para fornecer aos usuários as ferramentas adequadas e simples para armazenar dados confidenciais em elementos de armazenamento de uma forma transparente e segura. Seu uso no projeto HeMoLab tem grande importância, pois os dados do paciente trafegam por vários nós antes da execução.

Muitas vezes, *jobs* que demandam longo processamento exigem monitoramento e controle durante a sua execução. No HeMoLab, existem várias simulações que necessitam de horas de processamento. Usando o Watchdog [Bruno et al. 2009], é possível realizar o controle e monitoramento das tarefas usando os serviços da *Grid* de forma menos invasiva.

4. Projeto do Sistema Integador

Os principais desafios da portabilidade de um sistema de cluster local que usa MPI para um ambiente de *Grid* inclui a adaptação do trabalho de um sistema homogêneo para um ambiente heterogêneo, a criação de camadas de *software* que atuam como interface entre a aplicação e a infra-estrutura de *Grid* e o tráfego dos dados através de domínios usando o certificado de autoridade. Além disso, pesquisas são necessárias para encontrar o melhor número de processadores para a simulação hemodinâmica, paralelização de métodos matemáticos para as simulações numéricas e a segurança da informação.

O projeto do *software* é uma fase fundamental para se ter um bom sistema, pois a partir dele tem-se uma visão completa sobre o que se deve fazer, aplicando estratégias que melhor possam atender às necessidades do *software*. Através dele, é possivel transformar os resultados de análise de requisitos em documentos que podem ser interpretados pelos desenvolvedores do sistema.

5. Requisitos e os Casos de Uso

O modelo de caso de uso consiste em atores e casos de uso. Os Atores representam algo que interagem com o sistema, humano ou não. Um caso de uso é uma sequência de ações oferecidas pelo sistema, os quais interagem com um ator em particular. Em outras palavras, um caso de uso é o modo como o ator utiliza o sistema [Jacobson and Bylund 2000].

A figura 2 mostra o diagrama de caso de uso para os requisitos definidos anteriormente, logo após, são descritos os atores e os casos de uso do sistema responsável por fazer com que o HeMoLab possa utilizar uma *Grid* computacional para o cálculo das simulações.

5.1. Atores

Seguem os Atores que fazem parte do diagrama de Casos de Uso:

HeMoLab: *software* para simulação hemodinâmica que interage com o sistema integrador, a fim de executar o resolvedor de simulações de forma paralela em *Grid*.

Cluster remoto: máquinas pertencentes a um *site* que irão, efetivamente, executar o resolvedor numérico de forma paralela. Estas máquinas fazem interface com a *Grid* através de uma máquina denominada *Computer Element (CE)*, que é o *Front-end* do *cluster* remoto.

5.2. Casos de Uso

A seguir, são descritos os Casos de Uso - ações oferecidas pelo sistema, os quais se interagem com os atores:

Configurar JDL: para cada *job* a ser submetido, um arquivo com extensão jdl deve ser configurado. Esta funcionalidade é responsável por definir como será feita a interação com o ambiente de *Grid*.

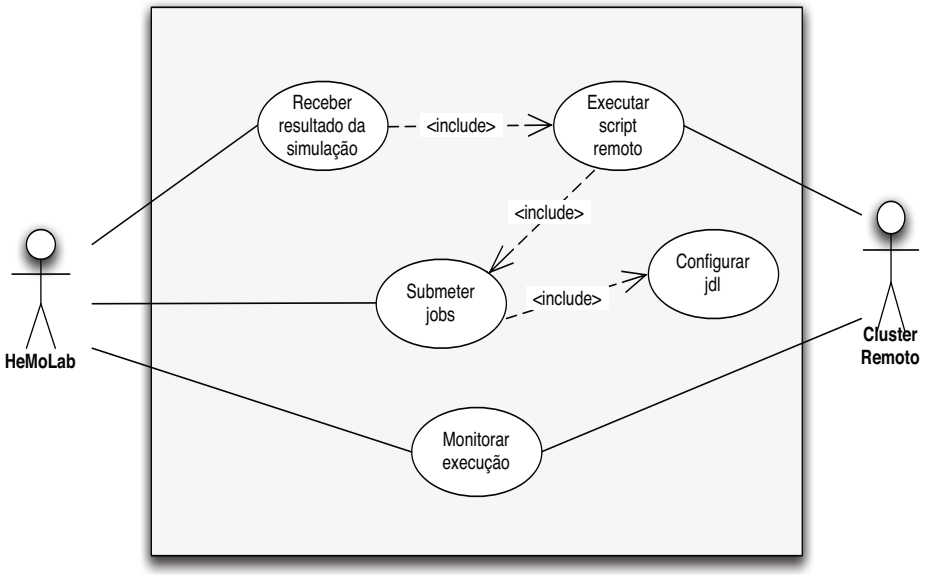

Figura 2. Diagrama de Casos de uso para o sistema

Cenário - definir forma de envio de arquivos: *i)* são informados quais arquivos devem ser enviados para as máquinas da *Grid* que têm como única função o armazenamento de arquivos; *ii)* são informados quais arquivos devem ser enviados diretamente ao CE.

Submeter jobs: funcionalidade responsável por enviar o *job* para o cálculo das simulações, além de enviar os arquivos de entrada para a execução do resolvedor numérico.
Cenário - submeter *job* e arquivos de entrada: *i)* arquivos de entrada são empacotados; *ii)* arquivos de entrada são criptografados; *iii)* os recursos computacionais são selecionados; *iv)* arquivos referentes ao modelo são enviados; *v) job* é submetido.

Executar script remoto: funcionalidade responsável por iniciar a execução do *script* remoto e mover os arquivos do SE para o CE para início da resolução da simulação numérica.
Cenário - executar arquivo remoto e iniciar execução do SolverGP: *i)* fazer *download* dos arquivos enviados ao SE; *ii)* enviar arquivos para os WorkerNodes; *iii)* receber arquivos de Saída.

Receber resultados da simulação: funcionalidade responsável por monitorar o estado de um *job* e fazer o download dos dados de saída, caso o *job* tenha terminado sua execução.
Cenário - recuperar dados de saída: *i)* monitorar estado do *job*: enviado, na fila, em execução, terminado ou falha; *ii)* fazer *download* dos arquivos de saída armazenados em um SE; *iii)* decriptografar arquivos de saída.

Monitorar execução: funcionalidade responsável por permitir ao usuário do HeMoLab, observar a convergência da simulação através da leitura do arquivo de *log*, gerado durante a execução do resolvedor numérico.
Cenário - observar saída produzida no arquivo de *log*: *i)* verificar se o *job* está em execução; *ii)* iniciar monitoramento de *job*.

6. Visões Arquiteturais

Segundo [Clements et al. 2002], uma arquitetura estabelece restrições ou um baixo fluxo de atividades, e estas atividades devem produzir artefatos que são inerentes à arquitetura, mas a arquitetura não define a implementação.

6.1. A Visão de Componentes e Implantação

Ainda segundo [Clements et al. 2002], uma visão de implantação mapeia processos para elementos de *hardware*: nós de processamento, canais de comunicação, memória e armazenamento. Os processos são os elementos de *software* geralmente utilizados neste tipo de visão.

A visão de implantação mostra quais processos são alocados para cada elemento de *hardware*. Uma camada de serviços foi criada entre os SolverGP e os serviços fornecidos pelo GLite Middleware. A figura 3 mostra o diagrama de componentes e implantação deste sistema:

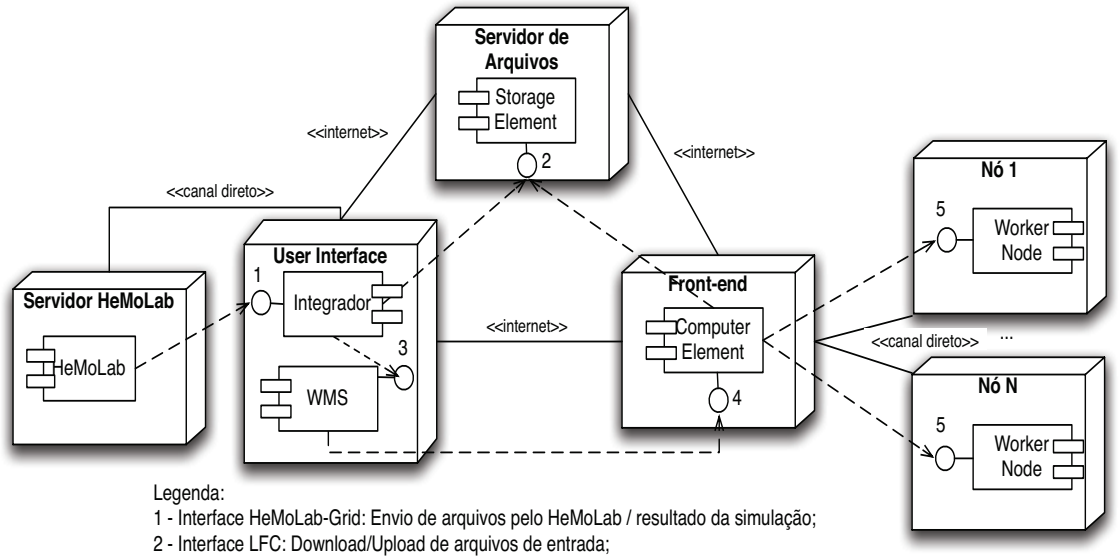

Figura 3. Diagrama de componentes e implantação

O SolverGP usa o paradigma de programação MPI, permitindo executar uma tarefa de forma paralela em máquinas pertencentes a um *cluster*. O componente Integrador, (instalado na User Interface), é responsável por pegar o programa executável SolverGP e designar um *site* da *Grid* para a execução. O Nó alocado, chamado Computer Element (CE), é responsável por fornecer acesso ao *cluster* remoto e distribuir as tarefas para as várias máquinas pertencentes a este *cluster*. O componente Integrador é o componente responsável pela interface com o CE, que usa um *script* remoto para configurar as máquinas do *cluster* (Worker Nodes), a fim de executar as tarefas de forma paralela. O Workload Management System (WMS) é um conjunto de componentes de *middleware Grid* responsável pela distribuição e gestão de tarefas através de recursos da *Grid*. O

Storage Element (SE) e LFC File Catalog (LFC) são responsáveis por: gerenciar o armazenamento (fornecendo espaço de armazenamento para arquivos) e serviços de transferência de arquivos grandes, e facilitar a disponibilidade e localização de arquivos de dados exigidos pelos *jobs* dos usuários, respectivamente. Os arquivos localizados no SE são acessível por usuários e aplicativos de qualquer lugar da *Grid*, várias réplicas de um arquivo podem ser armazenadas em diferentes locais e não podem ser alteradas (apenas removidas ou substituídas).

Resumindo, o componente Integrador é responsável por: *i)* criar uma *interface* entre o HeMoLab e o CE, a fim de iniciar o *script* remoto para a execução do SolverGP; *ii)* criptografar os arquivos de entrada e descriptografar os dados de saída através de uma chave privada usando o certificado digital do usuário; *iii)* acionar os recursos para monitoramento de *jobs*, exigidos pelo HeMoLab para acompanhar a simulação, a fim de saber se o problema está convergindo. *iv)* receber os arquivos de saída gerados remotamente, e enviá-los para exibição no HeMoLab.

6.2. A Visão de Módulos do Sistema

A visão de módulos do sistema mostra como a funcionalidade é mapeada para a implementação, identificando as diferentes unidades de código e como elas se interagem. A camada de Integração faz interface com a *Grid* através do *framework* disponibilizado pelo *middleware* gLite, que permite o uso do WMS (*Interface* do gLite com a *Grid* computacional). O módulo Executor remoto e o Resolvedor numérico, são escritos em linguagens Shellscript e Fortran, respectivamente. A Figura 4 mostra o diagrama de módulos para o sistema descrito.

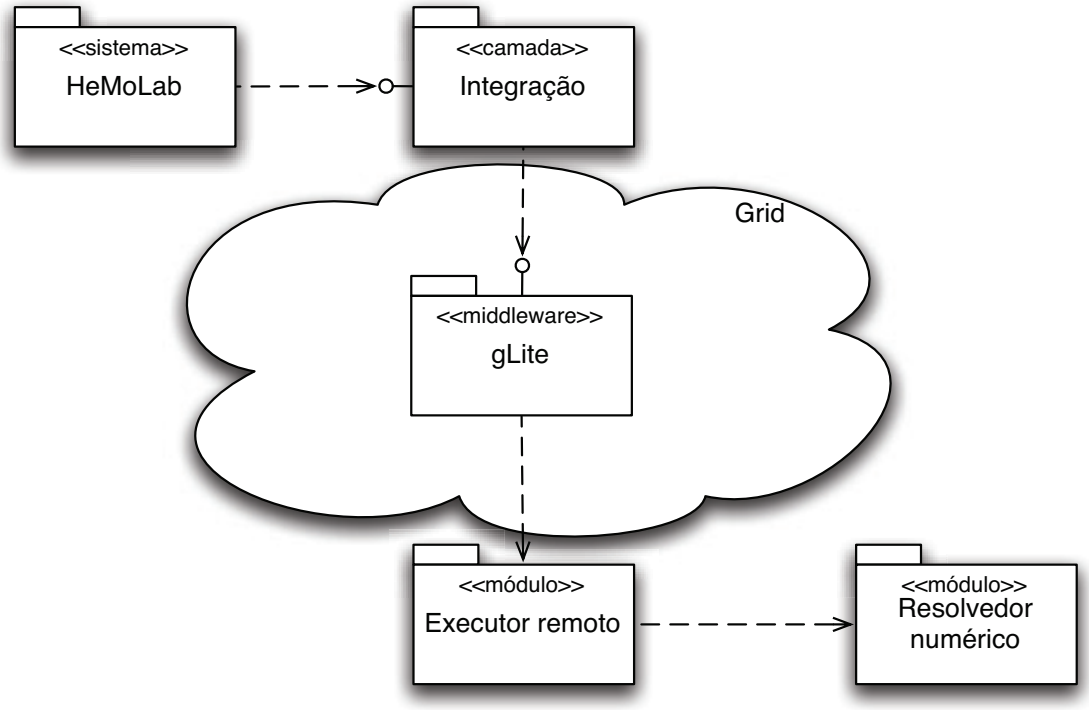

Figura 4. Diagrama de módulos do sistema

Vale a pena ressaltar aqui, que os módulos Executor remoto e Resolvedor numérico não estão inicialmente instalados em um nó remoto. Estes módulos são enviados através da *interface* que a camada de integração faz com o gLite. Apenas após determinar o *site* onde a simulação será executada, estes módulos são enviados. O executor remoto é o módulo capaz de disparar o Resolvedor numérico nas máquinas do *cluster* remoto.

6.3. A Visão de Execução

A visão de execução visa descrever os elementos de *hardware* e os processos do sistema associados ao *hardware*. Para o diagrama utilizado nesta visão, será mostrado o fluxo de execução de forma mais resumida. Para cada *interface*, será feita uma descrição mais detalhada. A Figura 5 mostra o diagrama de execução com o fluxo de execução da simulação numérica do HeMoLab e sua interação com o ambiente de *Grid*. O WMS é um conjunto de componentes de *middleware Grid* responsáveis pela distribuição e gestão de tarefas através de recursos da *Grid*. O LFC faz o mapeamento da estrutura fisica dos SE para nomes virtuais.

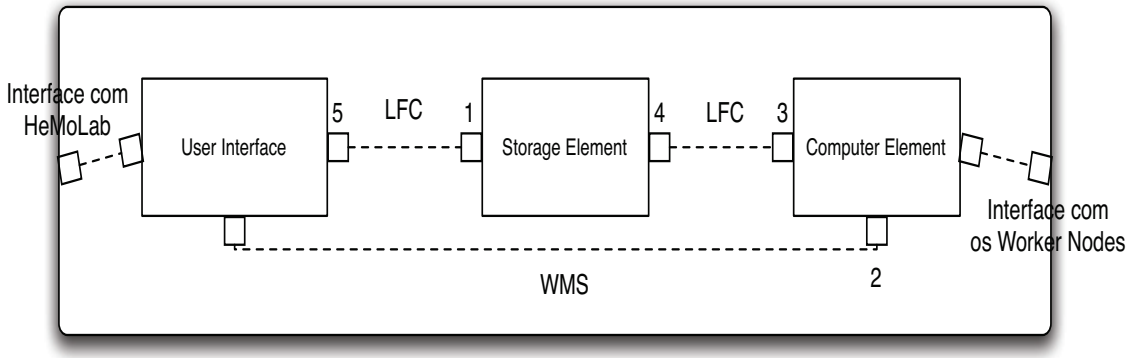

Figura 5. O fluxo de execução de uma submissão de *Job*

A User Interface (UI), considerada parte do WMS, é uma máquina de submissões, o ponto de entrada para a *Grid*. O componente Integrador está instalado na UI e contém o arquivo JDL, responsável por definir como será feita a interação com o ambiente de *Grid*. De acordo com a numeração no diagrama de execução (Figura 5), segue a explicação para o fluxo de execução:

1. Antes de submeter um *job* para processamento remoto, três arquivos devem ser armazenados no SE: *i) inputfiles* - arquivos que foram gerados pelo HeMoLab, exigidos para o resolvedor numérico; *ii) solverGP* - o executável do solverGP, responsável pela simulação numérica, *iii) libs* - bibliotecas do MPI e do HeMoLab, necessários para implementar as simulações numéricas. Os arquivos de entrada enviados para o SE são criptografados na *interface* do usuário, usando o Secure-store e a chave privada do usuário responsável pela submissão do *job*.

2. Um arquivo JDL é configurado para definir quais arquivos serão enviados para o CE através do WMS. Usando os parâmetros inputSandBox e outputSandBox, é possível definir quais arquivos serão enviados e recebidos pela UI. Três arquivos são enviados

para o CE através do WMS: *i) solver.sh - script* responsável por fazer o *download* dos arquivos armazenados no SE e iniciar a execução do SolverGP, descrito anteriormente; *ii) watchdog* - arquivos usados para permitir o acompanhamento *on-line* das etapas de execução da simulação; *iii) Secure-Store* - bibliotecas necessárias para decriptografar os arquivos de entrada.

3. O *script* remoto é executado, os arquivos são movidos do SE para o CE, a simulação numérica é iniciada e, finalmente, os dados de saída são gerados. No momento em que os dados de saída são gerados, podemos monitorar a convergência do processo de simulação, abrindo uma sessão Watchdog para acompanhar, em tempo real, os problemas que podem ter ocorrido na simulação numérica, ou verificar os arquivos de saída para análise imediata.

4. Os arquivos de saída são enviados para o SE e os arquivos de *log* são retornados para a UI através do WMS.

5. Os arquivos de saída são movidos do SE para a UI, para que sejam enviadas para exibição no HeMoLab.

7. Projeto Detalhado do Sistema

A especificação detalhada dos módulos será expressa por meio dos diagramas UML de classes e de sequência. Não será abordado o diagrama UML de pacotes, por se achar inadequado para a simplicidade que este diagrama seria.

7.1. Diagrama de Classes

Um diagrama de Classes descreve os tipos de objetos presentes no sistema e os vários tipos de relacionamentos estáticos existentes entre eles, mostram as propriedades e as operações de uma classe e as restrições que se aplicam à maneira como os objetos estão conectados [Fowler 2004].

Para o diagrama de classes, serão descritos: o propósito de cada classe, uma breve narrativa do funcionamento de cada classe, uma definição detalhada da *interface* de cada classe e suas dependências/restrições com outras classes.

A Figura 6 mostra o diagrama de classes para o sistema proposto:

Classe Jdl: classe que provê um mecanismo para armazenamentos das informações sobre o *job* que será submetido: *i) inputFiles:* caminho para os arquivos de entrada necessários para a resolução da simulação; *ii) outputFiles:* caminho para o arquivo onde deve ser escrito o resultado da simulação; *iii) blackList:* endereço de *sites* que não devem ser escolhidos para executar o resolvedor numérico; *iv) numberOfCores:* número de núcleos necessários para executar o resolvedor numérico; *v) jobType:* tipo do *job*; *vi) filter:* requisitos para execução do resolvedor numérico; *vii) fileToExec:* nome do arquivo que deve ser executado remotamente.

Para cada atributo da classe, existem os métodos *get* e *set* associados. Estes, são utilizados para armazenar e recuperar os valores dos atributos, descritos anteriormente. Um método, chamado *createJdl()*, é responsável por escrever os dados de cada atributo em um arquivo com extensão jdl.

Classe Glite: classe responsável por fazer *interface* com o *framework* do gLite, desta forma, é possível tratar os resultados antes do processamento no Integrador.

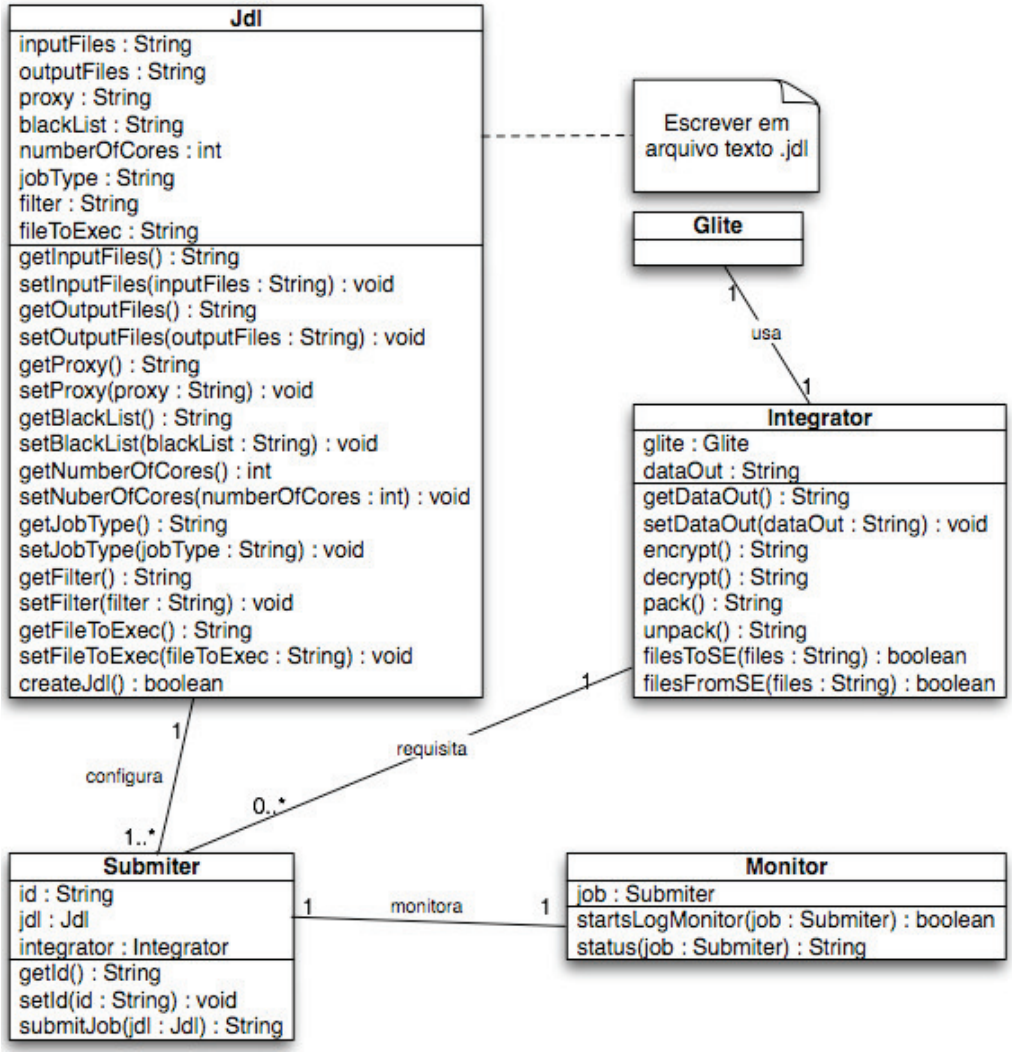

Figura 6. Diagrama de classes

Classe Integrator: classe responsável pela comunicação entre o gLite e as outras classes pertencentes ao sistema: *i) dataOut:* atributo que armazena o caminho no qual os arquivos recuperados da *Grid* estão armazenados. Atributos *get* e *set* estão associados a este atributo; *ii) encrypt() e decrypt():* métodos responsáveis por criptografar e decriptografar arquivos de entrada e saída; *iii) pack() e unpack()* métodos responsáveis por empacotar e desempacotar arquivos; *iv) fileToSE():* responsável por fazer o tráfego de dados para a *Grid*; *v) filesFromSE:* responsável por recuperar dados armazenados em nós da *Grid*.

Classe Submiter: classe responsável pela submissão de *jobs* e identificação dos *jobs* enviados ao gLite: *i) id:* atributo que armazena a identificação de um *job* submetido à *Grid*. Métodos *get* e *set* são associados a este atributo; *ii) submitJob():* a partir de um arquivo jdl configurado, o método submete o *job* associado.

Classe Monitor: classe responsável por iniciar a sessão para analisar a saída de *jobs* que estejam em execução: *i) startsLogMonitor():* a partir de um *job* em execução (*running*), este método permite que uma sessão seja aberta, a fim de verificar o arquivo

de *log* gerado pela simulação que está sendo processada; *ii) status():* responsável por verificar o estado de um *job* submetido ao gLite.

7.2. Diagrama de Sequência

O diagrama de sequências é uma representação comportamental, que indica como eventos provocam transições de objetos para objetos. Em suma, o diagrama de sequência é uma versão abreviada do caso de uso, uma representação de como os eventos causam fluxo de um objeto para outro como função do tempo [Pressman 2004].

A Figura 7 mostra o diagrama de sequências para o sistema abordado:

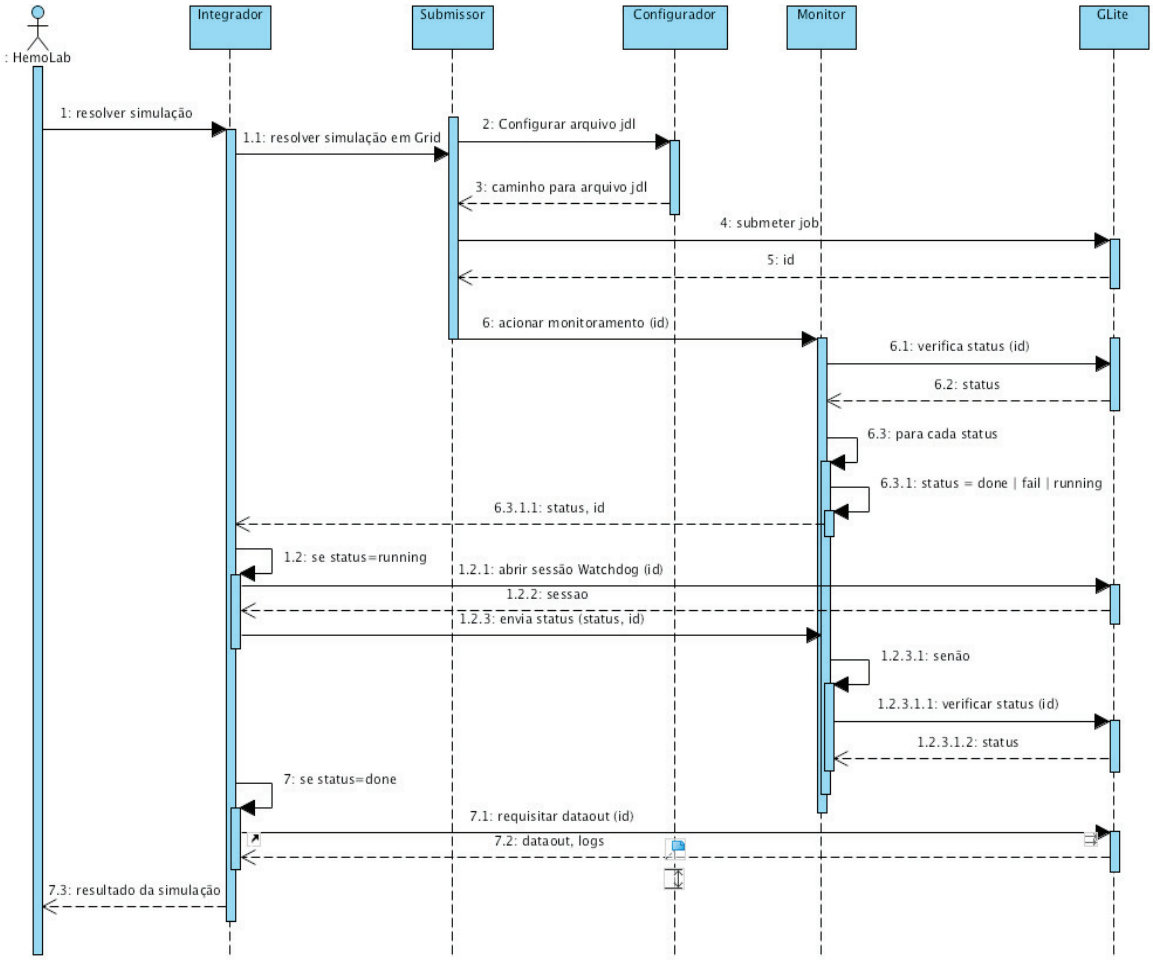

Figura 7. Diagrama de sequência

8. Implementação

Pretende-se implementar o integrador junto ao *software* HeMoLab, implementado com base no Paraview III [Moreland 2009]. Este, além de utilizar diversas classes próprias, utiliza a bilbioteca VTK e C++ como linguagem de programação principal. Como biblioteca de *interface* gráfica será utilizado Qt [Nokia]. Para acessar as funções do glite diretamente da aplicação, pretende-se utilizar a API C++ disponibilizada por este. Desta forma, pretende-se facilitar os processos de busca de recursos computacionais (que satisfaçam os requisitos da aplicação), submissão dos *jobs*, acompanhamento da simulação e, por fim,

busca de resultados. Uma vantagem do uso da API é que, os estágios acima citados, podem ser processados com um menor nível de controle interativo do usuário da aplicação, provendo assim um ambiente mais amigável e, ao mesmo tempo, que utiliza os recursos em *Grid* na etapa de computação de alto desempenho.

A Figura 8, mostra o botão que, ao ser acionado, levará o fluxo de execução para o *software* integrador, reponsável por fazer com que o resolvedor numérico seja processado em uma *Grid* Computacional usando o gLite Middleware.

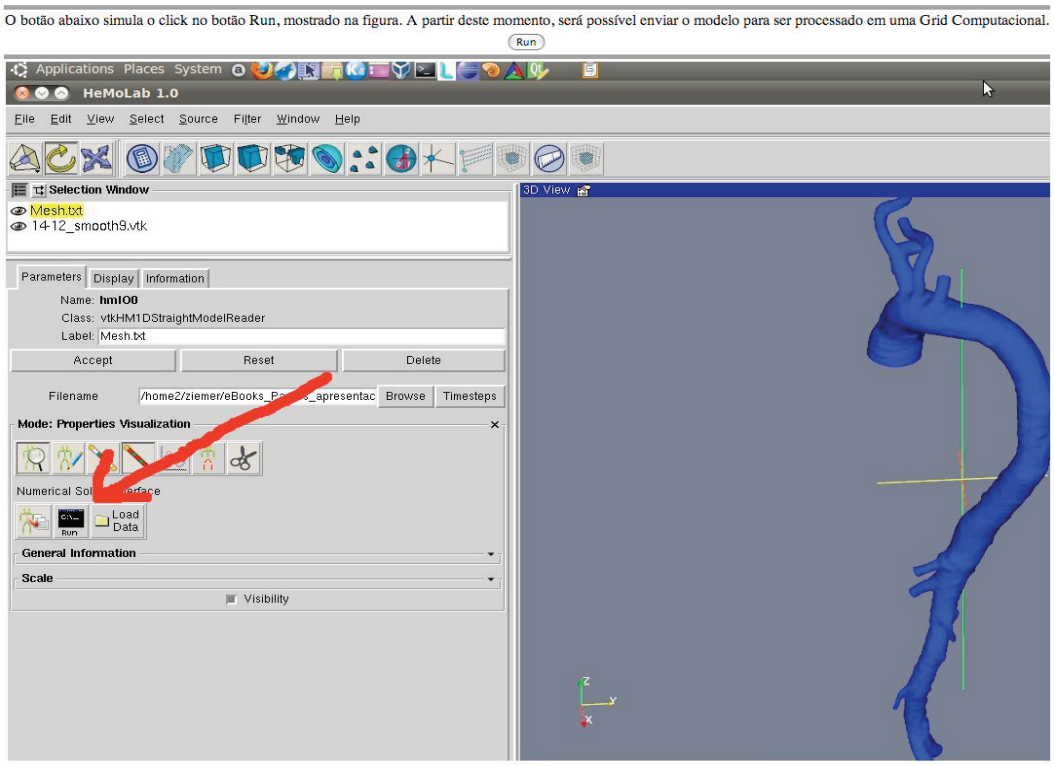

Figura 8. Tela do HeMoLab com o botão *Run*, que irá acionar o integrador

9. Conclusões e Trabalhos Futuros

Doenças relacionadas com o sistema cardiovascular humano, tais como doenças cardíacas coronarianas, são as principais causas de morte no mundo. Desta forma, modelar o comportamento deste sistema é vital para melhor compreendê-lo e pode ajudar na criação de novas terapias. O processo de modelagem pode fazer uso de modelos simplificados (1D), modelos com maior nível de detalhes (3D), e modelos acoplados (1D-3D). Esses tipos de modelos permitirão a realização de estudos de patologias cardiovasculares relacionadas, por exemplo, ao processo de envelhecimento da artéria aorta, aneurismas e estenoses, e simulação de procedimentos cirúrgicos, como a inserção do *stent*, a criação de revascularização, etc. No entanto, a simulação numérica de modelos 3D ou modelos acoplados têm mostrado que o tempo de processamento exigido é da ordem de semanas ou mesmo meses em modelos mais complexos. Desta forma, o uso de técnicas de computação de alto desempenho, como a computação em *Grid* é vital para o sucesso da adoção da ferramenta pela comunidade médica.

Este trabalho apresentou a modelagem conceitual do sistema integrador que permite com que um sistema de simulação hemodinâmica possa ser processado em uma *Grid* Computacional. Foram especificados os requisitos, a arquitetura e a correspondência entre os elementos descritos em ambas as especificações. A especificação de requisitos foi expressa em linguagem natural, além do diagrama UML de caso de uso. A especificação arquitetural contemplou os três principais tipos de visões arquiteturais: execução, módulo e implantação. Foi apresentado um projeto detalhado do sistema integrador, através de diagramas UML de pacotes, de classes, e de sequência, além de incluir um plano de testes descrevendo casos de teste unitário e de integração. Como trabalho futuro, espera-se a realização de simulações computacionais mais intensas para utilização de um número maior de processadores. Um aspecto muito importante é que o uso da computação em *Grid* pode ajudar a especificação do número ideal de processadores usados na resolução de um modelo, ou seja, fazer um estudo mais profundo da curva de *speedup* do SolverGP. Isso permitirá saber com antecedência a quantidade aproximada de núcleos que resolvem um modelo com um número específico de pontos em melhor tempo. Também será utilizada a API do gLite para a construção de um portal *Web* para acompanhamento dos *jobs* executados na *Grid*.

Durante este trabalho, várias lições foram aprendidas, entre elas podemos destacar: i) a necessidade de um projeto bem definido para tornar o *software robusto*; ii) o uso de ferramentas de projeto para ajudar na melhor visualização do produto final, afim de tornar cada vez menor o uso de refatoração.

Dentre as principais contribuições, podemos destacar a possibilidade de simular múltiplos *jobs* de forma paralela, um em cada *cluster* remoto, utilizando de forma gráfica a configuração para que isto seja possível. Finalmente, espera-se que o uso do aplicativo possa ajudar na difuzão do uso da ferramenta HeMoLab entre a comunidade de pesquisa médica brasileira, pois mais simulações podem ser feitas ao mesmo tempo. Espera-se que estas simulações também possam ser resolvidas em um tempo menor que o praticado atualmente.

Para mais detalhes sobre o projeto HeMoLab, veja [HeMoLab].

Referências

Blanco, P. J., Pivello, M. R., Urquiza, S. A., and Feijóo, R. A. (2009a). Building coupled 3d–1d–0d models in computational hemodynamics. In *1st International Conference on Mathematical and Computational Biomedical Engineering - CMBE2009*, Swansea, UK.

Blanco, P. J., Pivello, M. R., Urquiza, S. A., and Feijóo, R. A. (2009b). On the potentialities of 3d-1d coupled models in hemodynamics simulations. *Journal of Biomechanics*, 42(7):919–930.

Bruno, R., Barbera, R., and Ingra, E. (2009). Watchdog: A job monitoring solution inside the eela-2 infrastructure. In *Proceedings of the Second EELA-2 Conference*, Choroni, VE.

Burke, S., Campana, S., and et. al. (2009). *gLite User Guide*. `http://glite.web.cern.ch/glite/`.

Clements, P., Bachaman, F., Bass, L., Garlan, D., Ivers, J., Little, R., Nord, R., and Stafford, J. (2002). *Documenting Software Architecture – Views and Beyond*. Addison-Wesley.

Forum, M. P. (1994). Mpi: A message-passing interface standard. Technical report, Knoxville, TN, USA.

Fowler, M. (2004). *UML distilled: a brief guide to the standard object modeling language*. Pearson Education, Inc, 3rd edition.

Gropp, W., Lusk, E., and et. al. (2009). *MPICH2 - Message-Passing Interface (MPI) Implementation Users Guide*. Argonne National Laboratory, US. `http://www.mcs.anl.gov/research/projects/mpich2/`.

HeMoLab. Hemodynamics modeling laboratory. `http://hemolab.lncc.br`.

Jacobson, I. and Bylund, S. (2000). *The road to the unified software development process*. Cambridge University Press.

Mackay, J., Mensah, G. A., Organization, W. H., for Disease Control, C., and Prevention (2004). The atlas of heart disease and stroke - deaths from coronary heart disease. *World Health Organization, Geneva*, page 112.

Moreland, K. (2009). *The ParaView Tutorial - A Parallel Visualization Application*. Sandia National Laboratories. `http://www.paraview.org/`.

Nokia. Qt - cross-platform application and ui framework. `http://qt.nokia.com/products/`.

Pressman, R. S. (2004). *Software Engineering Software Engineering: A Practitioner's Approach, 6th Ed*. McGraw Hill, 6th edition.

Scardaci, D. and Scuderi, G. (2007). A secure storage service for the glite middleware. *International Symposium on Information Assurance and Security*.