Universidade Estadual de Campinas

Faculdade de Engenharia Elétrica e de Computação

Departamento de Microondas e Óptica

Simulador Eletromagnético em um Ambiente de Grades Computacionais

Autor: Igor José Ferreira de Freitas

Orientador: Hugo Enrique Hernández Figueroa

Co-Orientador: Marli de Freitas Gomes Hernández

Trabalho apresentado à Faculdade de Engenharia Elétrica e de Computação da UNICAMP como parte dos requisitos exigidos para obtenção do título de Mestre em Engenharia Elétrica.

Comissão Examinadora

Prof. Dr. Marli de Freitas Gomes Hernández (FT/UNICAMP) – Presidente

Prof. Dr. Antonio Romeiro Sapienza (DETEL/UERJ)

Prof. Dr. André Franceschi de Angelis (FT/UNICAMP)

Prof. Dr. Rui Fragassi Souza (Consultor)

Campinas, 08 de Setembro de 2010

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA E ARQUITETURA - BAE - UNICAMP

F884s

Freitas, Igor José Ferreira
Simulador eletromagnético em um ambiente de grades computacionais / Igor José Ferreira de Freitas. -- Campinas, SP: [s.n.], 2010.

Orientador: Hugo Enrique Hernández Figueroa; Marli de Freitas Gomez Hernández.
Dissertação de Mestrado - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Eletromagnetismo. 2. Simulação (Computadores). 3. Modelos geométricos. 4. Serviços web. 5. Computação em grade (Sistemas de computador). I. Hernández Figueroa, Hugo Enrique. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Título em Inglês: Electromagnetic simulator in a grid computing enviroment
Palavras-chave em Inglês: Electromagnetism, Simulation (Computer), Geometric modeling, Web services, Computational grids (Computer systems)
Área de concentração: Telecomunicações e Telemática
Titulação: Mestre em Engenharia Elétrica
Banca examinadora: Antonio Romeiro Sapienza , André Franceschi de Angelis , Rui Fragassi Souza
Data da defesa: 08/09/2010
Programa de Pós Graduação: Engenharia Elétrica

COMISSÃO JULGADORA - TESE DE MESTRADO

Candidato: Igor José Ferreira de Freitas

Data da Defesa: 8 de setembro de 2010

Título da Tese: "Simulador Eletromagnético em um Ambiente de Grades Computacionais"

Profa. Dra. Marli de Freitas Gomes Hernandez (Presidente): _____

Prof. Dr. Antonio Romeiro Sapienza: _____

Prof. Dr. André Franceschi de Angelis: _____

Agradecimentos

Agradeço primeiramente a Deus pelas conquistas e realizações e por ter me proporcionado todas as condições para realização deste trabalho.

Gostaria de agradecer imensamente o apoio e orientação do Prof. Hugo Enrique Hernández Figueroa, orientador, e a Profa. Marli de Freitas Gomez Hernández, co-orientadora, que, durante este tempo conduziram a orientação com objetividade, criticidade e amizade.

Um agradecimento aos meus amigos do DMO, Carmen Lúcia, Carlos Henrique, Gilliard Nardel, Kleucio Cláudio, Leonardo Ambrósio e Rafael Buck.

Gostaria de agradecer a toda minha família, em especial meus pais e meu irmão, meu tio Orlando, meus amigos de república Rafael Freitas e Vinicius Mommensohn.

Por fim, agradeço à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo apoio financeiro.

Dedico este trabalho
à minha mãe, Maria Lúcia;
ao meu pai, Aparecido;
e ao meu irmão
Marcos André.

Resumo

Este trabalho tem por objetivo desenvolver os estágios de Pré e Pós-processamento de um pacote de simulação eletromagnética, batizado de SSAR-BR, e também de um ambiente de grades computacionais que permitirá a resolução de problemas eletromagnéticos de grande porte.

O estágio de processamento já foi desenvolvido como parte de uma tese de doutorado pertencente ao grupo de pesquisa deste departamento e se baseia no método das Diferenças Finitas no Domínio do Tempo (FDTD). O estágio de Pré-Processamento envolve um modelador básico de duas e três dimensões para modelagem geométrica de dispositivos de telecomunicações (fontes radiantes, terminais móveis e antenas) e modelos biológicos (partes do corpo humano e cobaias). O estágio de Pós-Processamento, por outro lado, permite visualizar os resultados das simulações eletromagnéticas executadas, através de gráficos em duas e três dimensões. Estes dois estágios já foram desenvolvidos nas linguagens Java e OpenGL.

O ambiente computacional paralelo está baseado no conceito de Grade Computacional (do inglês *Grid*), isto é, numa infra-estrutura de *software* e *hardware* que visa à integração de recursos computacionais geograficamente dispersos, interconectados através de uma rede, com o objetivo de compartilhar tais recursos sob determinadas políticas de acesso, formando um ambiente computacional escalável, de baixo custo e de alto desempenho. A camada de *software* localizada entre a aplicação do usuário e o sistema operacional responsável por controlar o acesso aos recursos da Grade Computacional, é chamada de middleware. O middleware adotado é o UNICORE 6, que se caracteriza por fornecer acesso aos recursos da Grade Computacional de forma transparente e segura, seguindo padrões abertos definidos pela comunidade OGF (*Open Grid Forum*).

Abstract

The aim of this work is to develop the Pre and Post-Processing stages from an electromagnetic simulation suite, called SSAR-BR, and also a grid computational environment, which permits the execution of large electromagnetic problems.

The processing stage, based in the Finite Difference Time Domain (FDTD) method, has been already developed as part of a doctorate thesis made by a researcher from this department. The Pre-processing stage involves a fundamental modeling tool capable to model telecommunications devices (radiant sources, mobile terminals and antennas) and biological models (human body parts and guinea-pig). On the other hand, the Post-Processing stage permits visualizing the electromagnetic simulations results in a two and three dimensions way. Both stages were developed in Java and OpenGL languages.

The parallel computational environment is based on Grid Computing paradigm which consists in *software* and *hardware* infra-structure that integrates dispersed computational resources interconnected across a network. Its purpose is to share such resources under an access control policy and establishing a scalable, low-cost and high performance computational environment. The middleware is the layer between the user and operation systems responsible to manage the Grid resources and, for this work, the UNICORE 6 was adopted. The mainly reasons were its safe and transparent access to the distributed environment and, most important, it follows the open patterns from OGF (Open Grid Forum) community.

Sumário

Sumário i
Lista de Figuras iii
Capítulo 1 7
Introdução 7
1.1 Motivação 7
1.2 Objetivos 8
1.3 Estrutura do Trabalho 8
Capítulo 2 10
Grades Computacionais 10
2.1 Computação Orientada a Serviços 10
2.2 Aplicações de Grades Computacionais 12
2.3 Desvantagens de Grades Computacionais 13
2.4 Middleware UNICORE 6 15
2.5 Grades Computacionais 22
Capítulo 3 29
Simulador Eletromagnético – Modelador Básico 29
3.1 Introdução 29
3.2 Arquitetura 31
Capítulo 4 43
Pós-Processamento 43
4.1 Introdução 43
4.2 Arquitetura 43
Capítulo 5 52
Validação e Resultados 52
5.1 Validação Inicial do FDTD 3D e SAR 52
5.2 Validação do SSAR-BR com medições feitas no CPqD 56
5.2.1 Montando a Cena de Simulação 56
5.2.2 Teste de SAR sem o recipiente 61
5.3 Teste Validação Experimental e Simulador 66

5.4 Vantagens na utilização do software SSAR-BR 71
Capítulo 6 74
Conclusões 74
6.1 Trabalhos Futuros 76
Referências Bibliográficas 77

Lista de Figuras

Figura 1. Arquitetura geral do UNICORE 6 [26] 17
Figura 2. Requisições no *Gateway* UNICORE 6 [37] 20
Figura 3. Grades Computacionais para Disponibilização de Recursos 23
Figura 4. Esquema de *Cluster* LE-45 24
Figura 5. Encapsulamento do método FDTD 3D 25
Figura 6. *Workflow* para uma simulação através do UNICORE 6 25
Figura 7. Gerenciador de Arquivos – GPE File Manager 26
Figura 8. Determinando tempo de simulação através do GPE-Client 27
Figura 9. Simulação remota em andamento 28
Figura 10. Arquitetura geral do SSAR-BR 31
Figura 11. Esquema de camadas do Modelador Básico e Gerenciador 32
Figura 12. Visão Geral do Módulo Modelador Básico e Gerenciador 34
Figura 13. Árvore de Objetos da Cena de Simulação 35
Figura 14. Modelos geométricos representados por objetos Geometria 36
Figura 15. Diagrama de Classes – Estrutura dos Objetos Modelados e/ou Importados 37
Figura 16. Tela do Gerenciador / Modelador Básico 41
Figura 17. Transição de Objeto Vetorial para Objeto Malhado 42
Figura 18. Interface do módulo de Pós-Processamento 44
Figura 19. Diagrama de Classes de Alto Nível do Módulo de Pós-Processamento 45
Figura 20. Diagrama de Sequência – Gráfico SAR 46
Figura 21. Superfície SAR gerada pelo Pós-Processamento 46
Figura 22. Propagação do Campo Eletromagnético 47
Figura 23. Diagrama de Sequência – Filme 2D 48
Figura 24. Imagem de ressonância magnética tratada e redimensionada em 256x256 49
Figura 25. Arquivo de IRM Visualizado no *Software*. 50
Figura 26. Representação do setup experimental para analisar SAR [48]. 53
Figura 27. Modelo computacional geométrico do pyrex discretizado em um seção transversal para o FDTD 3D. Considerou-se o domínio computacional total com 165x165x165 células. 53
Figura 28. Domínio computacional modelado para os testes SAR. 54
Figura 29. Rampa Senoidal para alimentação do dipolo. 54
Figura 30. a) Medido [2], b) Simulado FDTD [48] c) Simulado com o FDTD 3D SSAR-BR 55
Figura 31. Tela de abertura do SSAR-BR onde o usuário seleciona se deseja criar novo projeto ou abrir algum existente. 57
Figura 32. Tela inicial padrão do SSAR-BR para novos projetos. 57
Figura 33. Modelo do recipiente de testes SAR modelado no 3D Studio® e importado para o SSAR-BR. 58
Figura 34. Especificar o material da geometria inserida ou importada. 59
Figura 35. Cadastrar nova fonte eletromagnética para simulações. 59
Figura 36. Inserindo fonte eletromagnética cadastrada na cena de simulação. 60
Figura 37. Malha do recipiente elipsóide modelado pela equipe do CPqD**.** 60
Figura 38. Planos de teste em destaque, da base para o topo um passo de 2mm**..** 61
Figura 39. Comparativo do máximo da SAR em simulações sem o recipiente. O plano 1 está a 2 mm da base. 62

Figura 40. Comparativo do máximo da SAR em simulações sem o recipiente. O plano 2 está a 4 mm da base. 62
Figura 41. Comparativo do máximo da SAR em simulações sem o recipiente. O plano 3 está 6 mm da base. 63
Figura 42. Comparativo do máximo da SAR em simulações sem o recipiente. O plano 4 está 8 mm da base. 63
Figura 43. Comparativo da superfície de SAR no plano 1 (a) superfície experimental e (b) superfície simulada. 64
Figura 44. Comparativo da superfície de SAR no plano 2 (a) superfície experimental e (b) superfície simulada., no plano 3 (c) superfície experimental e d) superfície simulada e no plano 4 (e) superfície experimental e (f) superfície simulada. 65
Figura 45. Recipiente de testes no formato STL (triangularizado). 66
Figura 46. Três visões no Matlab© da malha de testes: (a) visão superior do recipiente (b) visão lateral da bacia (c) visão frontal onde se visualiza o dipolo. 68
Figura 47. Valores comparativos de SAR considerando o recipiente e o líquido de teste no a) plano1 (b) plano2 (c) plano 3 e (d) plano 4. 69
Figura 48. Comparativos das superfícies de SAR experimental e simulado para: (a) plano1 experimental, (b) plano 1 simulado, (c) plano2 experimental, (d) plano 2 simulado, (e) plano 3 experimental, (f) plano 3 simulado, (g) plano 4 experimental e (h) plano 4 simulado. 71
Figura 49. Etapas da simulação eletromagnética 72

Lista de Acrônimos

AJO - *Abstract Job Object*

BAS – *Batch Environment Services Activity Service*

DMO – Departamento de Microondas e Óptica

DRMAA - *Distributed Resource Management Application* API

DXF - *AutoDesk Drawing Interchange Format*

FDTD – *Finite Diference Time Domain*

FTS - *File Transfer Service*

GUI – *Graphical User Interface*

HPC-P - *High Performance Computing Basic Profile*

IDB – *Incarnation Database*

IRM – Imagem de Ressonância Magnética

JMS - *Job Management Service*

JSDL - *Job Submission Description Language*

MVC - *Model-View-Controller*

OGBES - *Open Group Batch Environment Services*

OGSA - *Open Grid Service Architecture*

OGSA-BES - *Open Grid Services Architecture – Basic Execution Services*

RFS - RUS-*Factory Service*

RGBA - *Red, Green, Blue e Alpha*

RUS - *Resource Usage Service*

SAGA - *Simple API for Grid Applications*]

SAML - *Security Assertion Markup Language*

SAR – *Specific Absortion Rate*

SMS - *Storage Management Service*

SOA – *Service Oriented Architecture*

SOAP - *Simple Object Access Protocol*

SOC – *Service-Oriented Computing*

STL - *Standard Tessellation Language*

TCP/IP – Transfer Control Protocol / Internet Protocol

TLS - *Transport Level Security*

TRS - *Target System Registry Service*

TSF - *Target System Factory*

TSI – *Target System Interface*

TSS - *Target System Service*

UAS - UNICORE *Atomic Services*

UDDI - *Universal Description, Discovery and Integration*

UNICORE - *Uniform Interface to Computing Resources*

URF - *Usage Record Format*

VO – *Virtual Organizatiion*

WS – *Web Services*

WS-RF - *Web Services Resource Framework*

XACML - *eXtensible Access Control Markup Language*

XML - *eXtensible Markup Language*

XNJS - *Enhanced Network Job Supervisor*

XUUDB – *Extensible* UNICORE *User Database*

Capítulo 1

Introdução

1.1 Motivação

Atualmente, a demanda por sistemas computacionais de grande porte com alta complexidade tem aumentado, tanto no âmbito das empresas quanto no meio acadêmico. Espera-se que haja um desenvolvimento rápido, de baixo custo e com alta qualidade destes sistemas [1]. Esta demanda também ocorre no eletromagnetismo computacional devido aos métodos numéricos utilizados empregarem algoritmos paralelos em simulações mais complexas [2]. Para contornar este problema, grande parte das instituições de pesquisa possui sistemas de *clusters* que interconectam diversos computadores através de uma tecnologia de rede onde a comunicação entre eles é feita através de bibliotecas de passagem de mensagens baseadas nos protocolos da arquitetura TCP/IP. Esta abordagem é adotada devido à boa escalabilidade e ao baixo custo quando comparado a computadores paralelos.

Apesar destas vantagens, em muitos casos, os sistemas de *clusters* não são capazes de atender a demanda por processamento. Isto ocorre por diversas razões técnicas relacionadas à segurança do sistema, controle e gerenciamento de processos e interoperabilidade. Assim, a Computação em Grade (*Grid Computing*) emerge como uma tecnologia capaz de interligar e disponibilizar um conjunto de recursos computacionais distribuídos.

Portanto, a primeira motivação para este trabalho é implementar e configurar um ambiente de Grades Computacionais, através do *middleware* UNICORE 6, com o objetivo de integrar, compartilhar e disponibilizar recursos computacionais (*software* e *hardware*) desenvolvidos pelo Departamento de Microonda e Óptica (DMO) da Faculdade de Engenharia Elétrica e de Computação (FEEC) da UNICAMP.

Os componentes de *software* necessários à realização de simulações eletromagnéticas envolvem, basicamente, três etapas: Pré-Processamento, Processamento e Pós-Processamento. Enquanto a etapa de Processamento possui diversas implementações envolvendo os métodos

FDTD [4-6] (Diferenças Finitas no Domínio do Tempo) e FEM (Método dos Elementos Finitos) [2, 3], as demais ainda são dependentes de *software* comerciais com alto custo de aquisição e, além disso, muitos *software* não possuem uma integração satisfatória com as ferramentas já desenvolvidas pelo departamento. Esta dependência gera a segunda motivação para este trabalho, a carência de soluções de Pré e Pós-Processamento otimizadas para as atividades de pesquisa realizadas pelo grupo de pesquisa do DMO.

1.2 Objetivos

Este trabalho possui dois objetivos principais: o primeiro trata da implementação de um ambiente de Grades Computacionais, baseada no *middleware* UNICORE 6, para Simulação Eletromagnética. Este ambiente será responsável pela disponibilização dos recursos de *hardware* e *software* existentes no DMO, primando pela usabilidade, segurança e alto desempenho.

O segundo objetivo consiste em desenvolver um Modelador Básico para o Pré-Processamento, responsável pela modelagem geométrica para dispositivos a serem simulados, como, por exemplo, antenas, terminais móveis e modelos biológicos (humanos e cobaias), e um módulo de Pós-Processamento para visualização dos resultados obtidos em simulações eletromagnéticas realizadas pelo método FDTD 3D. Estes componentes serão integrados ao *software* SSAR-BR, descrito no Capítulo 3.

1.3 Estrutura do Trabalho

No Capítulo 2 são introduzidos a terminologia e os conceitos básicos pertinentes às Grades Computacionais e o *middleware* UNICORE 6, itens necessários para o entendimento do ambiente computacional em que os componentes do Simulador Eletromagnético estão inseridos.

A arquitetura e funcionalidades dos módulos Modelador Básico e de Pós-Processamento são apresentados nos Capítulos 3 e 4, respectivamente. Além disso, no Capítulo 5 têm-se os resultados e validações do SSAR-BR com resultados experimentais obtidos no laboratório de

ensaios para radiações não ionizantes do CPqD. O Capítulo 6 apresenta as conclusões sobre os componentes desenvolvidos, sua contribuição e algumas sugestões para trabalhos futuros.

Capítulo 2

Grades Computacionais

Este capítulo apresenta uma visão geral sobre as tecnologias envolvidas em um ambiente de Grade Computacional e o *middleware* UNICORE 6 utilizado neste trabalho.

A primeira definição de Grades Computacionais foi feita por Foster e Kesselman [7] como sendo uma infra-estrutura de *software* e *hardware* provendo segurança, consistência, acesso universal e com baixo custo aos recursos computacionais.

Apesar de esta definição ser bastante abrangente, outras definições surgiram na literatura [8] ressaltando que um ambiente de Grade Computacional é um *framework*, no qual existem diversas camadas responsáveis pelo oferecimento de serviços para acessar e gerenciar seus recursos. Outras definições também abordaram o fato destes recursos estarem disponíveis em uma rede distribuída de computadores de alto desempenho sendo compartilhados por usuários internos e externos à rede.

Neste trabalho, a definição de Grades Computacionais segue a sugerida por [7], que a descreve como um sistema atuante na integração, virtualização e gerenciamento de serviços e recursos em um ambiente distribuído, heterogêneo e que interliga um conjunto de usuários definidos como Organizações Virtuais (VO) em domínios organizacionais e tradicionalmente administrativos (organizações reais).

2.1 Computação Orientada a Serviços

A Computação Orientada a Serviços (SOC – *Service-Oriented Computing*) pode ser definida como um novo paradigma em que o desenvolvimento de componentes de *software* de forma rápida, com baixo custo e reusabilidade são elementos para a composição de serviços em

um ambiente heterogêneo [9]. O termo "serviços" é entendido como um produto de *software* atômico, independente de outros serviços, e que contém uma interface, permitindo seu uso por um cliente através de uma rede.

Este conceito é concretizado através da implementação do SOA (*Service-Oriented Architecture*) ou Arquitetura Orientada a Serviços, a qual implica disponibilizar sistemas de "busca", utilização e combinação de serviços interoperáveis para garantir os processos e estratégias de uma organização.

2.1.1 *Serviços Web*

Os Serviços *Web* (WS) são definidos como um padrão para integrar aplicações *Web* utilizando tecnologias abertas como XML (*Extensible Markup Language*), SOAP (*Simple Object Access Protocol*), WSDL (*Web Services Definition Language*) e UDDI *(Universal Description, Discovery and Integration*) através da Internet. Neste caso, o XML é utilizado para classificar os dados, o WSDL para descrever os serviços e o UDDI para listar quais estão disponíveis. Assim, baseados em padrões abertos, os Serviços *Web* oferecem uma interoperabilidade universal com foco na troca de mensagens e seus componentes de *software* são fracamente acoplados [13] [15].

2.1.2 *Open Grid Service Architecture (OGSA)*

O GGF (*Global Grid Forum*) adotou o OGSA (*Open Grid Services Architecture*) como o padrão para Grades Computacionais. OGSA consiste em um padrão de desenvolvimento aberto (*Open*) que fornece interoperabilidade. O termo "*Grid*" faz referência a integração, virtualização e gerenciamento de serviços e recursos em um ambiente distribuído e heterogêneo. Além disso, é considerado um padrão Orientado a Serviços, pois oferece funções ou módulos fracamente acoplados, alinhando-se ao padrão *Web Services*, amplamente aceito pelo meio acadêmico e comercial [8].

Portanto, uma Grade Computacional é formada por uma estrutura composta de diversos componentes, dentre os quais, são:

- **Gerenciamento de Organizações Virtuais**: módulo responsável por identificar quais nós e usuários pertencem à determinada Organização Virtual.
- **Gerenciamento de Serviços e Alocação de Recursos**: cada aplicação encapsulada em uma grade possui requisitos para seu funcionamento, como, por exemplo, número de processadores, quantidade de memória RAM, bibliotecas instaladas e características de *software* e *hardware*. Assim, este módulo é responsável por descobrir algum nó (elemento da grade) compatível com os requisitos e, então, gerenciá-los.
- **Gerenciamento de Tarefas (*Job*):** módulo responsável por gerenciar e executar as tarefas enviadas pelos usuários aos nós da grade.

Além destes componentes principais há outros tipos de gerenciadores nas áreas de segurança, armazenamento de dados e gerência de pagamentos por uso dos serviços da grade.

Todos estes componentes tem um fluxo de interações entre si constante. Por exemplo, o componente de Gerenciamento de Tarefas recebe um pedido de serviço de um usuário, faz uma requisição ao componente Gerenciamento de Serviços e Alocação de Recursos, que, por sua vez, procura no Gerenciamento de Organizações Virtuais por um nó que atende a estes requisitos. Assim, dependendo deste fluxo de interações, uma grade pode entrar em colapso facilmente.

Devido a esta complexidade, a solução encontrada foi estabelecer uma padronização da arquitetura de um sistema de Grade Computacional definindo uma interface comum para cada tipo de serviço. Esta padronização foi efetuada pelo *Global Grid Forum* definindo um conjunto de normas para os componentes acima citados [10].

2.2 Aplicações de Grades Computacionais

Atualmente, as Grades Computacionais são consideradas uma das tecnologias fundamentais para a próxima geração em desenvolvimento de *software* [11, 12], pois se caracteriza como uma arquitetura de compartilhamento de recursos computacionais de forma padronizada em ambientes heterogêneos. Desta forma, pode-se utilizar Grades Computacionais em três diferentes aspectos, descritos a seguir.

- **Aproveitamento de *hardware* subutilizado:** quando o usuário submete um serviço à grade, este pode ser executado nas máquinas previamente configuradas de forma transparente ao usuário. Desta forma, todo poder computacional que existe em uma organização é aproveitado. Estes recursos podem ser computadores disponíveis em uma rede local, *clusters*, impressoras, armazenamento de dados entre outros [13].
- **Aproveitamento de *software* subutilizado:** atualmente existem diversos *software* desenvolvidos em trabalhos de iniciação científica, mestrado, doutorado entre outras pesquisas que, depois de concluídas, não são utilizadas ou compartilhadas com os demais pesquisadores. Em um ambiente colaborativo como a Grade Computacional, estes recursos são disponibilizados e reutilizados de forma prática pelos usuários.
- **Computação de Alto Desempenho:** muitas aplicações científicas necessitam de alto poder computacional. Isto implica na utilização de *clusters* e supercomputadores e essas aplicações podem fazer uso dos recursos oferecidos pelas Grades Computacionais, agregando máquinas da própria instituição ou até mesmo espalhadas pelo mundo [14].
- **Serviços sob demanda:** cada usuário da grade terá um *login* de acesso. Desta maneira é possível determinar quando e quais recursos serão oferecidos a estes usuários. Esta disponibilização de serviços pode ser feita de forma remota para outras instituições provendo também um acesso seguro.

2.3 Desvantagens de Grades Computacionais

Apesar dos benefícios das Grades Computacionais, esta tecnologia apresenta alguns desafios e limitações inerentes ao seu ambiente de operação (Web) que são citadas a seguir:

- **Descoberta de serviços:** todo serviço disponibilizado por uma Grade Computacional é hospedado em um servidor com seu respectivo endereço de

acesso. Caso este sofra alguma alteração, os Clientes deste serviço devem ser avisados.

- **Confiabilidade:** protocolos como o HTTP, utilizado pelas Grades Computacionais, não garantem a entrega das mensagens ao destino especificado.

- **Segurança:** todas as camadas de segurança para acesso restrito podem conter falhas aumentando-se a vulnerabilidade do sistema.

- **Transações:** transações de longa duração devem ser gerenciadas pelas Grades Computacionais.

- **Escalabilidade:** Deve-se empregar um mecanismo de balanceamento de carga satisfatório para possibilitar o aumento da capacidade do serviço que pode ser mensurada através da quantidade de acessos e tempo de resposta.

- **Escalabilidade:** Balanceamento de carga torna-se um desafio às Grades Computacionais, pois, prover flexibilidade de serviço com o objetivo de aumentar a capacidade de processamento, quantidade de acessos e diminuir o tempo de resposta são itens complexos de se alcançarem.

- **Desempenho:** Devido ao uso de protocolos como SOAP e HTTP, que empregam o XML como formato de dados, têm-se um aumento no volume de dados que trafegam pela Internet, aumentando-se, desta forma, o tempo de resposta. Além disso, muito processamento é utilizado na tradução das informações XML e extração dos dados SOAP.

- **Disponibilidade:** para utilizar-se das Grades Computacionais têm-se como pré-requisito o servidor e a própria Internet estarem disponíveis.

2.4 Middleware UNICORE 6

O UNICORE (*Uniform Interface to Computing Resources*) teve sua primeira versão desenvolvida em 1997 com o objetivo de prover aos usuários dos centros de supercomputadores alemães uma forma de acesso segura e com boa usabilidade aos recursos computacionais existentes [15]. Da mesma forma que o *middleware* Globus® Toolkit [16], o UNICORE foi desenvolvido antes do conceito de computação em grade (*Grid Computing*) e foi financiado pelo Ministério da Educação e Pesquisa do governo alemão.

Para manter uma arquitetura aberta e de acordo com os padrões estabelecidos pela OGSA, o UNICORE teve seus principais componentes reestruturados no projeto *European UniGrids* [17]. Como resultado, este projeto obteve uma maior interoperabilidade entre os principais *middlewares* como o Globus, gLite e CROWNGrid.

Nas seções seguintes é feita uma avaliação e justificativa da escolha do UNICORE 6 para compor um ambiente de Grades Computacionais responsável pela integração e disponibilização dos componentes de *software* desenvolvidos neste trabalho.

2.4.1 *Por que usar UNICORE 6 ?*

Devido à grande quantidade de *middlewares* no meio acadêmico e comercial [18] a escolha por um determinado sistema torna-se uma tarefa complexa. Neste trabalho foram estabelecidos alguns critérios, alcançados pelo UNICORE 6, como compatibilidade com outros *middlewares*, facilidade na configuração, segurança e suporte.

Outra característica que incentiva sua adoção é a utilização do UNICORE 6 por grandes *players* da indústria e instituições de pesquisa. Segue abaixo alguns exemplos de projetos que fazem uso deste *middleware*.

- **DESHL (*DEISA Services for the Heterogeneous management Layer*):** provê aos usuários uma interface tanto em modo gráfico como em console para gerenciamento de *jobs* e dados em um sistema de grade utilizando o *middleware* UNICORE. Faz uso de padrões como SAGA (*Simple API for Grid Applications*),

JSDL (*Job Submission Description Language*) e também do OGBES (*Open Group Batch Environment Services*) [19, 20].

- **D-Grid Integration Project (DGI):** este projeto faz parte de uma iniciativa alemã chamada D-Grid. O DGI busca oferecer suporte a uma infra-estrutura baseada em grades para aplicações científicas (*e-Science*). Os parceiros deste projeto visam à integração de aplicações de diferentes institutos de pesquisa [21].

- **Chemomentum**: este projeto visa melhorar o uso do UNICORE em aplicações de bioinformática adaptando e incluindo novos recursos de *workflow* [22].

- **OMII-Europe**: tem como foco facilitar o desenvolvimento e a portabilidade de um conjunto comum de aplicações para disponibilização de serviços a um grande número de *middlewares*. Construindo um forte acoplamento entre os mais aceitos, entre eles o UNICORE, Globus e gLite [23].

- **A-WARE**: provê acesso aos recursos de um ambiente de grade com foco na usabilidade através de uma interface *Web* baseada no *EnginFrame* e *GridSphere*. Este projeto possui cinco parceiros de quatro países europeus, Itália, Alemanha, Reino Unido e França. [24]

- **VIOLA:** *Vertically Integrated Optical Testbed for Large* Applications in DFN - voltado para teste de equipamentos de redes ópticas e arquitetura de redes. O UNICORE é utilizado como o *middlware* deste projeto para oferecer uma base para aplicações avançadas (pilotagem e visualização remota e realidade virtual) através das altas taxas de transferências proporcionadas por este tipo de rede [25].

2.4.2 *Arquitetura*

A arquitetura do UNICORE 6 tenta satisfazer um conjunto de requisitos de diversas áreas da computação, como segurança, usabilidade e configuração da infra-estrutura [26]. Deste modo, sua arquitetura abrange uma série de características como, uso de padrões conhecidos, autenticação e autorização robustas, garantia da integridade dos dados, cooperação com *firewalls* e necessidade de privilégios mínimos para os componentes do sistema.

A Figura 1 ilustra a arquitetura do UNICORE 6 configurada para duas organizações virtuais (VO) sob um *Gateway*. Nas seções seguintes é feita uma breve descrição de cada componente.

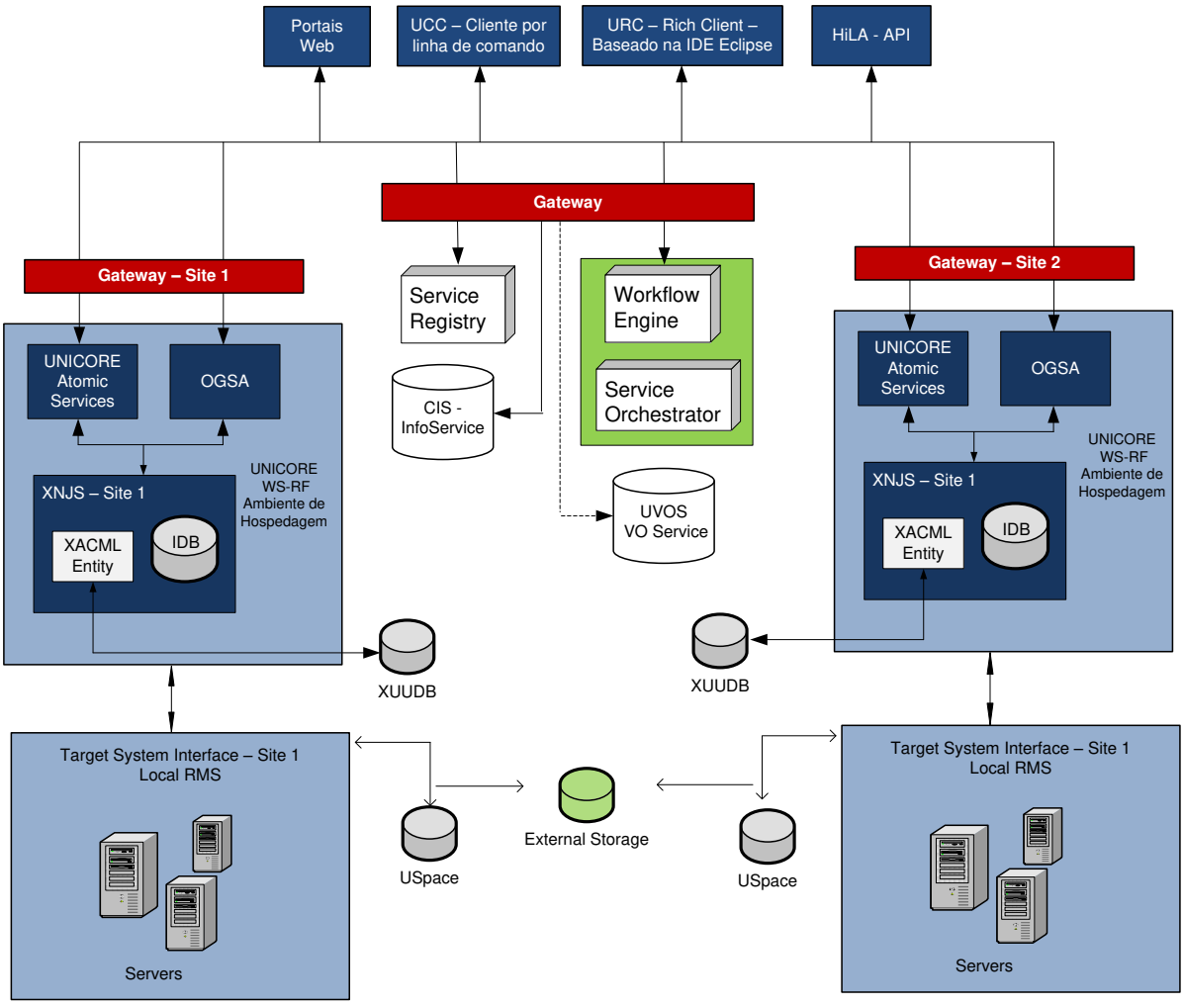


Figura 1. Arquitetura geral do UNICORE 6 [26]

2.4.3 *Atomic Services*

Este componente consiste em cinco *Web Services* estáveis que fornecem submissão de serviços (*Job Submission*) e fácil acesso aos recursos de armazenamento e transferência de arquivos. O mais importante deles é o *Target System Service* (TSS) que provê acesso ao *Target System WS-Resource*, no qual é responsável por estruturar os recursos computacionais que a grade possui.

Desta forma, é possível extrair informações detalhadas da grade como, por exemplo, quantidade de processadores, quantidade de memória, espaço de armazenamento e plataforma do sistema. O *Target System Factory* (TSF) é usado para criar o *Target System WS-Resource* seguindo uma implementação do WS-RF (*Web Services Resource Framework*) para o padrão fábrica (*Factory Pattern*).

O principal objetivo do TSS é disponibilizar os serviços da Grade através da linguagem JSDL, assim, evita-se o uso de linguagens proprietárias como AJO (*Abstract Job Object*), utilizada no UNICORE 5 [27]. Já a submissão de *jobs* via TSS consiste na criação do *Adressing Endpoint Reference* que pode ser usado para controlar os *jobs* através do JMS (*Job Management Service*).

O suporte a *data staging* dos *jobs* JSDL são feitos através do *Storage Management Service* (SMS) acessando os recursos de armazenamento do *WS-Resource* que representa um diretório remoto de onde os dados devem ser transferidos. A transferência é realizada pelo *File Transfer Service* (FTS) que utiliza os padrões OGF para transferência de dados através dos métodos *Random ByteIO* ou *Streamable ByteIO*.

2.4.4 *Workflow Engine*

O termo *workflow* pode ser definido como uma modelagem específica de um processo. Assim, uma sequência de tarefas ou atividades, representadas por um processo, pode ser definida em um *workflow* para a realização de determinada atividade. No UNICORE 6 o suporte à sistemas de *workflow* consiste em uma arquitetura de duas camadas, *Workflow Engine* (motores de *workflow*) e Serviços de Orquestração. Além da escalabilidade, esta arquitetura proporciona a utilização de diferentes motores de *workflows,* seguindo o paradigma da Arquitetura Orientada a Serviços.

2.4.5 *XNJS*

O XNJS (*Enhanced Network Job Supervisor*) pode ser definido como um gerenciador de *jobs* sendo o núcleo do UNICORE 6, pois provê recursos de armazenamento, transferências de arquivos e gerenciamento dos serviços. Este componente pode ser acessado através da interface proprietária UAS (UNICORE *Atomic Services*), descrita na seção acima, ou pelo conjunto de interfaces baseados em padrões abertos como o OGSA-BES (*Open Grid Services Architecture – Basic Execution Services*) [28] e HPC-P (*High Performance Computing Basic Profile*) [29] utilizados para criar, monitorar e controlar os *jobs*.

2.4.6 *Target System Interface*

O TSI é responsável pela transcrição de comandos efetuados a partir da grade para o sistema operacional no qual o UNICORE 6 está instalado. Assim, todo gerenciamento de *jobs* e transferência de arquivos é realizado por este componente que é compatível com a maioria dos sistemas *batch* atuais como Torque [30], LoadLeveler [31], LSF [32], SLURM [33] e OpenCCS [34]. Além disso, a versão do TSI para o padrão DRMAA (*Distributed Resource Management Application* API) [35, 36] possui uma interface para interoperabilidade entre o sistema *batch* e o próprio TSI.

2.4.7 *UNICORE Gateway*

O *Gateway* é a primeira camada do UNICORE 6 garantindo a autenticação do usuário e comunicação segura entre cliente e servidor provendo informações sobre os recursos disponíveis em cada site. De acordo com a Figura 2 o usuário envia uma requisição ao *Gateway*, este, por sua vez, se comunica com o servidor de destino abrindo apenas uma porta no *Firewall* [37].

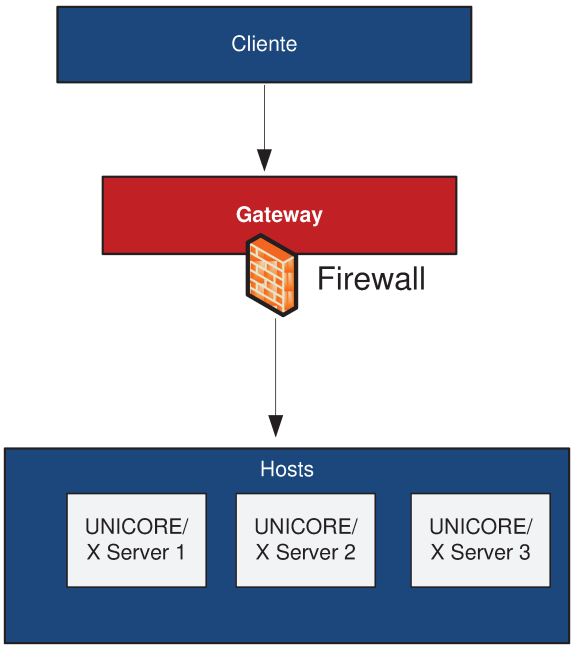


Figura 2. Requisições no *Gateway* UNICORE 6 [37]

2.4.8 *UVOS - UNICORE VO Services*

Este componente é uma alternativa ao XUUDB, provendo um sistema de autorização de usuários (*login*) a uma VO (Organização Virtual), neste caso, o UNICORE VO *Services* utiliza o padrão SAML (*Security Assertion Markup Language*). Mais informações sobre esta linguagem pode ser encontrada em [38].

2.4.9 *Registry Service Interface*

Este componente é responsável por disponibilizar de forma centralizada, todos os serviços e recursos disponíveis no UNICORE 6. Desta forma, o usuário consegue obter informações de quais recursos estão disponíveis para cada tipo de serviço. Estes são solicitados através do *Target System Registry Service* (TRS) que faz uso do *Service Groups*, um conjunto de especificações definidos pelo WS [39].

2.4.10 *OGSA – Basic Execution Service*

Como mostrado na Figura 1, o OGSA – BES *(Basic Execution Services)* define uma interface para WS para criação, monitoração e controle dos *jobs*. Estes *jobs* podem ser entendidos como entidades computacionais, processos em sistemas operacionais como UNIX ou Windows, *Web Services* ou programas paralelos (atividades) dentro de um determinado ambiente. Uma descrição detalhada pode ser encontrada em [40].

2.4.11 *OGSA – Resource Usage Service*

O *Usage Record Format* (URF) e o OGSSA (RUS) *Resource Usage Service* visam gerenciar e disponibilizar, de forma padronizada, as informações da base de usuários e utilização dos recursos na grade. Esta padronização, baseada em arquivos XML, permite a execução de uma série de serviços sobre um ambiente de grade, como por exemplo, gerenciamento de contas de usuários, cobrança por uso da grade, monitoramente de uso dos recursos e otimizações de desempenho. Informações detalhadas podem ser encontradas em [41].

2.4.12 *Incarnation Database - IDB*

Trata-se de um banco de dados responsável pelo mapeamento de um *job* descrito pela linguagem JSDL para *jobs* executáveis reais (processos nos sistemas operacionais). Além disso, as informações sobre aplicações disponíveis e características dos recursos da grade são gravadas neste banco de dados.

2.4.13 *Segurança*

O UNICORE 6 é baseado no certificado de segurança X.509 que utiliza o TLS (*Transport Level Security*). O TLS é responsável pela integridade, privacidade e autenticação na camada de transporte entre duas aplicações na Internet. A integridade garante que a troca de chaves encriptografadas se realiza apenas entre as duas partes envolvidas. A privacidade provê a criptografia do processo deixando-a ilegível a terceiros. A autenticação checa a identidade das partes envolvidas.

Além disso, a linguagem SAML é usada para reforçar diferentes modelos de políticas de autorização em conjunto com as políticas de segurança baseadas no XACML *(Extensible Access Control Markup Language)*, que se trata de uma linguagem de política de controle de acesso desenvolvida em XML.

2.5 Grades Computacionais

Um ambiente de Grade Computacional, descrito nas seções anteriores, traz inúmeras vantagens na disponibilização de recursos de *software* e *hardware* existentes em uma organização, neste caso, o DMO. Atualmente o departamento possui um *Cluster* do tipo *Beowulf* [36] já configurado, oferecendo uma vasta gama de aplicativos numéricos para variadas aplicações de eletromagnetismo computacional. Desta forma, é apresentada na Figura 3 uma visão de como este ambiente de Grades pode ser interligado aos módulos de Pré e Pós-Processamento desenvolvidos neste trabalho, como também disponibilizar acesso aos demais *software* existentes e que estão em desenvolvimento pelo DMO.

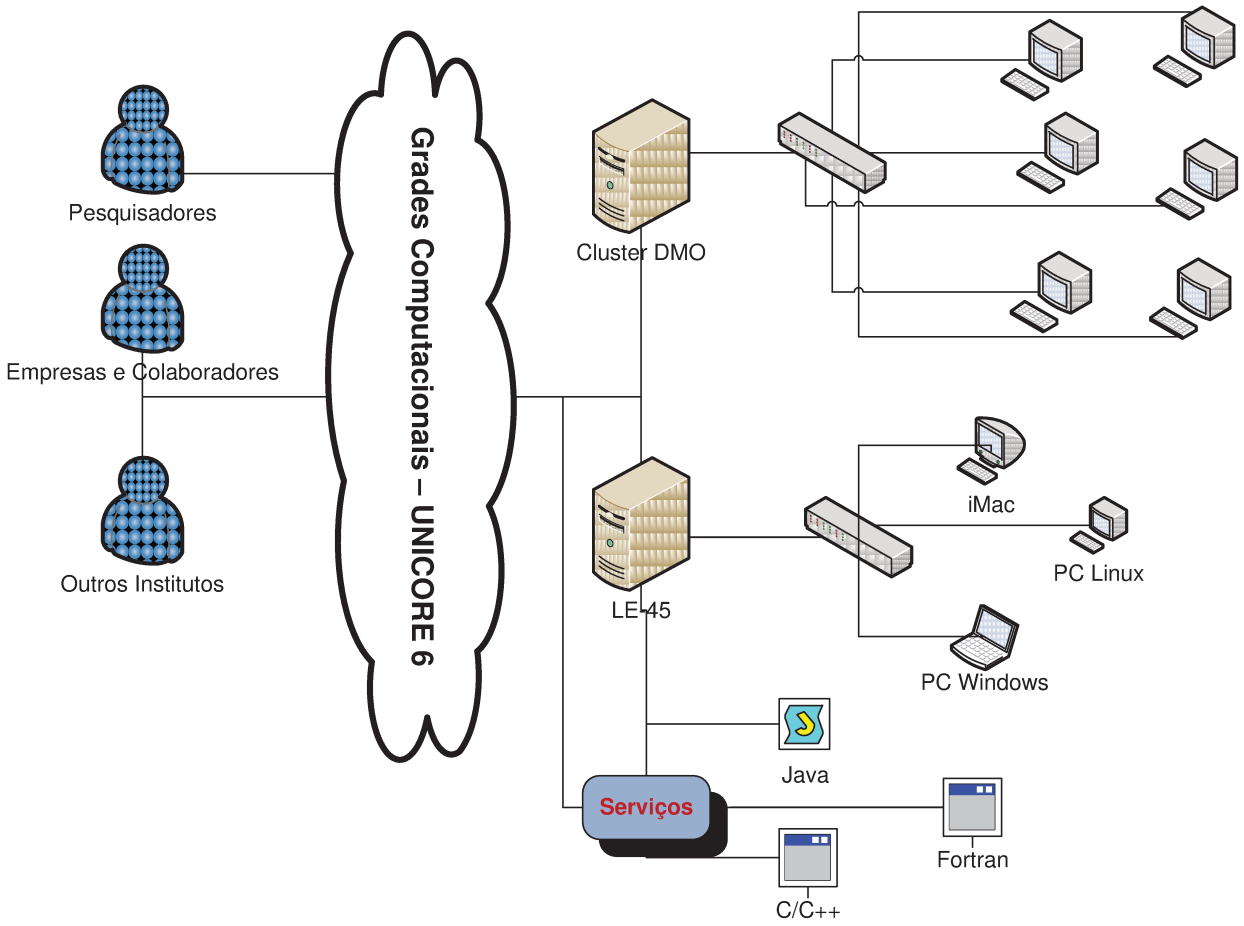


Figura 3. Grades Computacionais para Disponibilização de Recursos

Para concretizar este ambiente de Grade, foi utilizado o *middleware* UNICORE 6, descrito na Seção 2.3. Sua instalação foi efetuada no Laboratório LE-45, vinculado ao DMO, e sua configuração é visualizada na Figura 4.

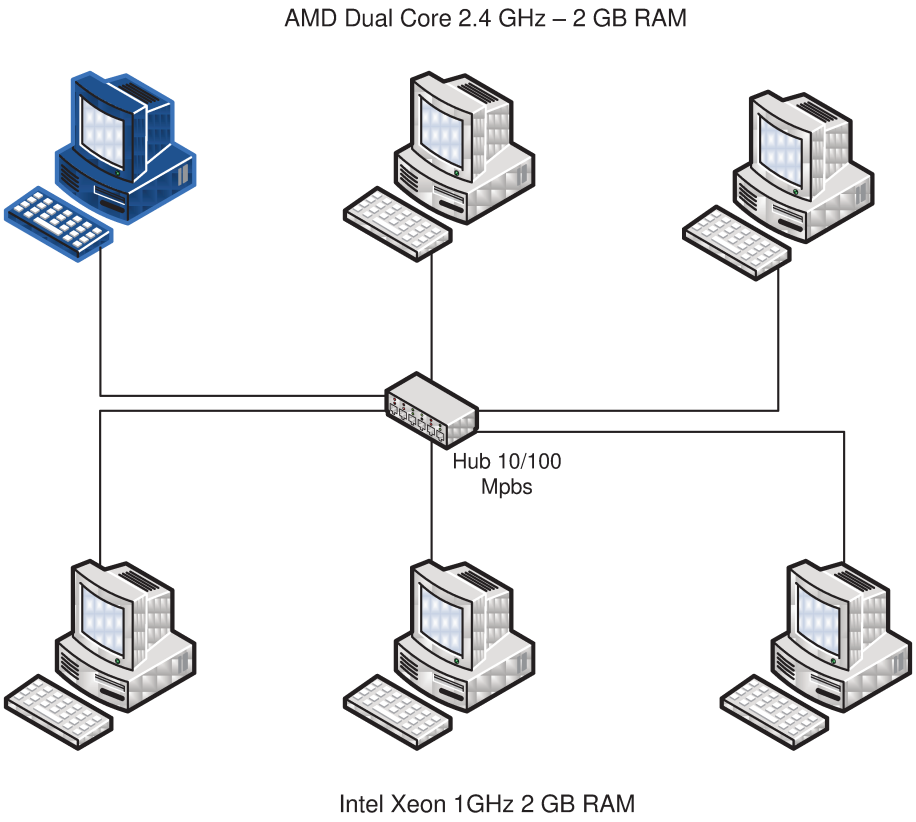


Figura 4. Esquema de *Cluster* LE-45

Na Figura acima, têm-se em destaque o ícone do microcomputador em azul, no qual se refere ao servidor onde foi instalado o *middleware* UNICORE 6 e que contém o método numérico FDTD 3D devidamente encapsulado. Este método foi desenvolvido pelo DMO, que de acordo com a arquitetura do *software* apresentada na Seção 3.1 que se refere ao módulo de Processamento.

Para encapsular as aplicações já existentes, é editado o arquivo "simpleidb", no formato XML, que se encontra na pasta de instalação do UNICORE 6, ("UNICORE_Quickstart\unicorex\conf\simpleidb"), e neste arquivo, conforme mostra a Figura 5, é inserido o endereço do arquivo binário, seja na plataforma Linux ou Windows.

```
<!-- FDTD3D -->
  <idb:IDBApplication>
    <idb:ApplicationName>FDTD3D</idb:ApplicationName>
    <idb:ApplicationVersion>1.0</idb:ApplicationVersion>
    <jsdl:POSIXApplication
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix">
   <jsdl:Executable>userPath/structure_PVC/fdtd3d</jsdl:Executable>
  </jsdl:POSIXApplication>
</idb:IDBApplication>
```

Figura 5. Encapsulamento do método FDTD 3D

Após a instalação e configuração dos arquivos, o fluxo de atividades, exemplificado na Figura 6, consiste em utilizar o *software* Simulador Eletromagnético nas fases de Modelagem e Definição de Materiais e Geração de Malha.

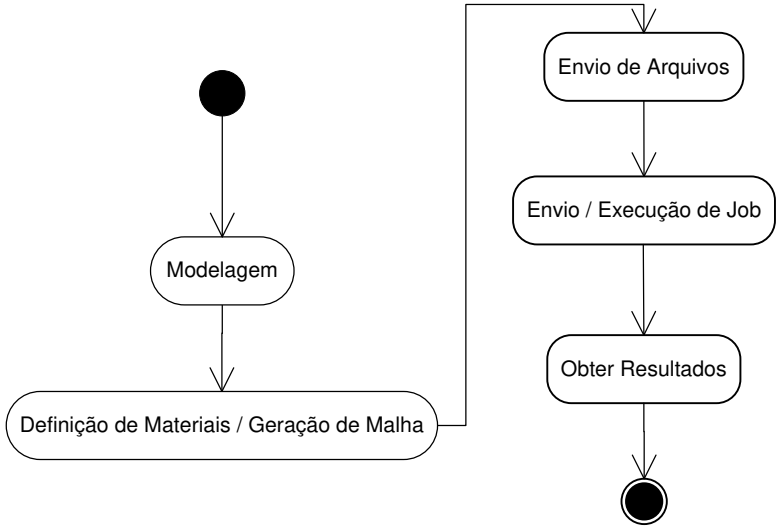


Figura 6. *Workflow* para uma simulação através do UNICORE 6

Após a geração do arquivo de malha, que contém o domínio computacional a ser simulado, o usuário deve fazer o envio destes arquivos ao ambiente de *Cluster*. Neste caso, é utilizado o *software* **GPE Application Client** em sua versão *File Manager*, desenvolvido pela Intel [46] de Arquivos do UNICORE 6 para o envio dos arquivos necessários à simulação, conforme é mostrado na Figura 7.

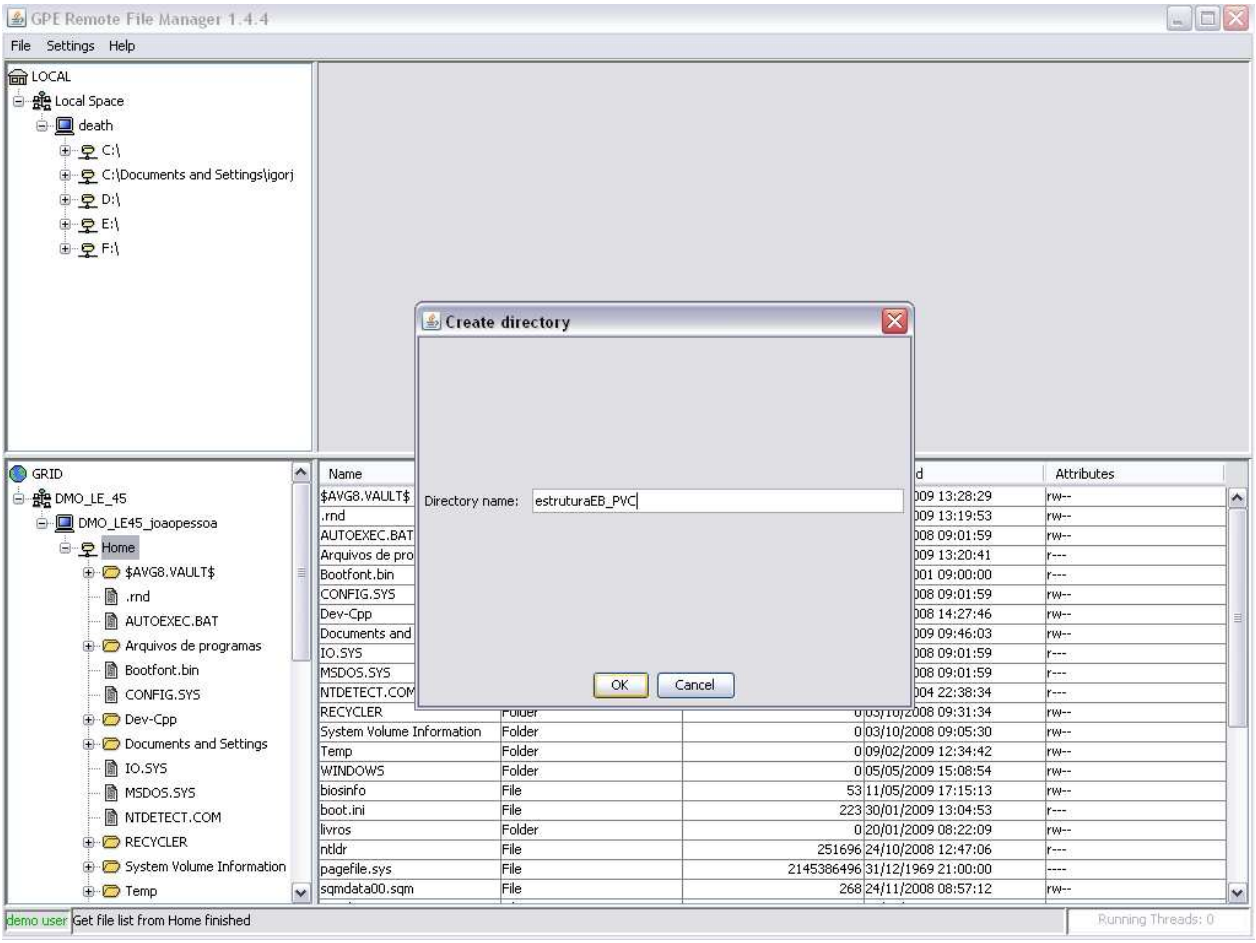


Figura 7. Gerenciador de Arquivos – GPE File Manager

Para esta simulação foi criada a pasta "estruturaEBG_PVC" e, em seguida, enviados os arquivos de entrada, "br_fdtd_config.isar", "br_fdtd_config_pml.isar", "br_fdtd_electromagneticsources.isar", "br_fdtd_fields.isar", "br_fdtd_materials.isar", "br_fdtd_mesh.isar", "br_fdtd_reports.isar" gerados pelo Simulador Eletromagnético; em seguida, foi criada a pasta "Resultados" e o arquivo "TotalTempo.txt".

Após o envio dos arquivos o próximo passo é enviar o pedido de execução de *Jobs;* esta tarefa é executada através do **GPE Application Client** em sua versão genérica, conforme é mostrado na Figura 8.

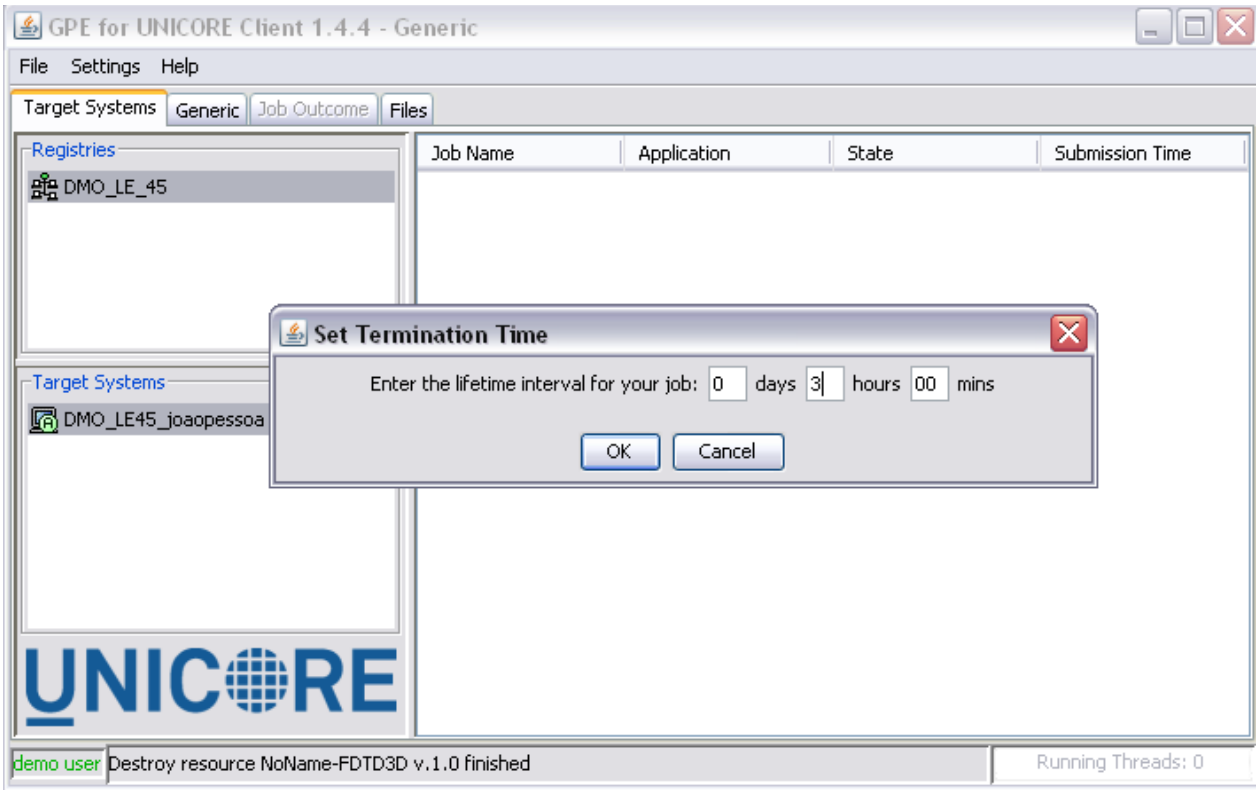


Figura 8. Determinando tempo de simulação através do GPE-Client

Alguns testes foram realizados em microcomputadores localizados no Laboratório LE-45; porém, para validar a utilização deste ambiente remotamente, foi instalado o **GPE Application Client** em uma estação de trabalho localizada fora da rede da UNICAMP. Esta abordagem pode ser visualizada através da Figura 9, onde o microcomputador que está rodando o *job* enviado, localizado no LE-45 é acessado remotamente via Logmein [47] e sua tela é capturada para que seja possível conferir se o *job* foi realmente enviado, e neste caso, através do Gerenciador de Tarefas do *Windows*, o arquivo binário "fdtd3d.exe" mostra-se em execução.

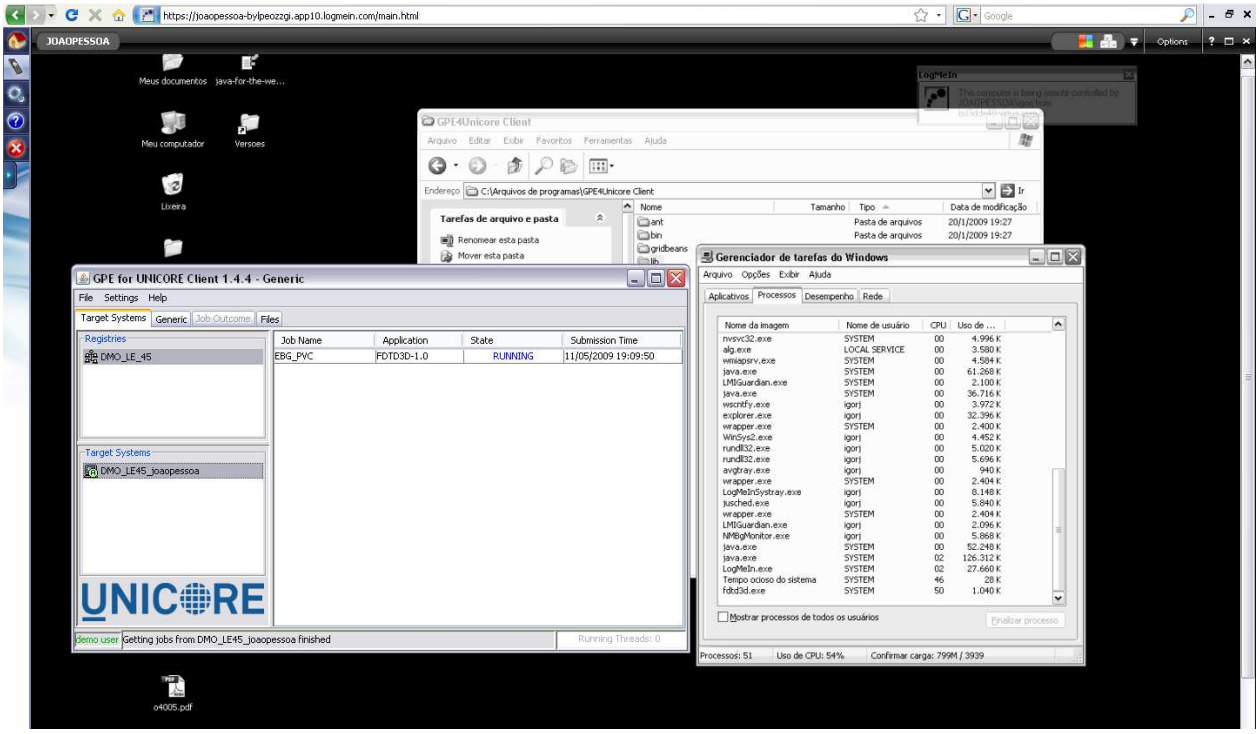


Figura 9. Simulação remota em andamento

Após o término da simulação os arquivos de resultados podem ser transferidos da máquina remota para a máquina cliente através do **GPE Application Client** na versão *File Manager*, da mesma forma que foram enviados os arquivos de entrada para simulação.

Capítulo 3

Simulador Eletromagnético – Modelador Básico

3.1 Introdução

A simulação de campos eletromagnéticos é composta por três etapas principais, Pré-Processamento, Processamento e Pós-Processamento. O Pré-Processamento é responsável pela modelagem geométrica dos dispositivos e estruturas a serem simulados. O Processamento é responsável pela execução do método numérico e, o Pós-Processamento, apresenta os resultados obtidos através da geração de gráficos em duas e três dimensões.

Neste projeto, foram desenvolvidos dois módulos, Pré-Processamento e Pós-Processamento. Os demais módulos apresentados no diagrama de pacotes na Figura 10, que compõe o *software* SSAR-BR, foram desenvolvidos por outros pesquisadores do DMO. Neste capítulo é apresentado o módulo Pré-Processamento, já para os demais módulos, é feita uma breve descrição de suas funções.

Módulo "Gerador de Malha"

Módulo responsável pela geração de malha tridimensional para o método numérico FDTD através de algoritmo de cruzamento de raios [41]; assim, este módulo faz um mapeamento dos objetos vetoriais tridimensionais que compõem a cena de simulação.

Módulo "*Processamento*"

Módulo responsável pela execução do método numérico FDTD através dos dados procedentes do módulo de Pré-Processamento, para análise da propagação dos campos eletromagnéticos.

Módulo "Importadores"

Módulo responsável pela importação de modelos geométricos em arquivos do tipo STL (*Standard Tessellation Language*) e DXF (*AutoDesk Drawing Interchange Format*).

Módulo "Mediador"

Este módulo é responsável pela comunicação central entre todos os módulos. Ele reflete o padrão de projeto "mediador" [42], que se utiliza do princípio de delegação para implementar os comandos necessários para fazer o aplicativo funcionar, conforme os eventos produzidos pelo usuário.

Módulo "Utils"

Trata-se de um pacote que contém as classes denominadas "utilitárias", estas podem ser utilizadas por um ou mais módulos do *software* SSAR-BR. Como exemplo têm-se as classes: ***Geometria*** , ***Material*** (define o tipo de material eletromagnético de cada geometria), ***Malha*** (Representa uma malha importada dentro de uma cena de simulação).

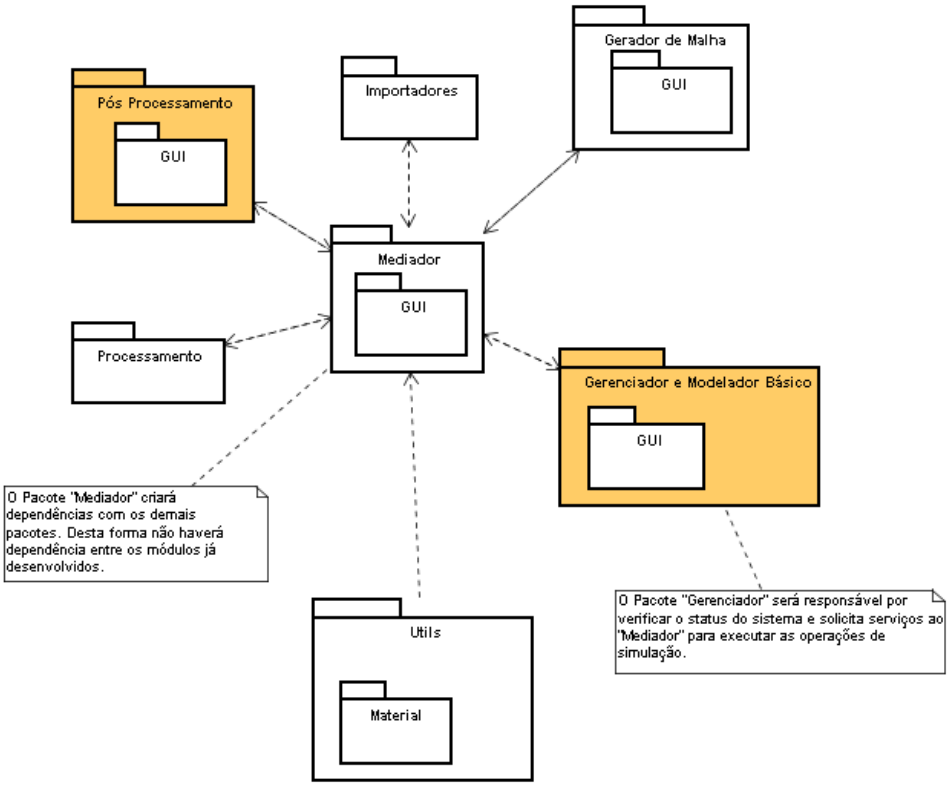


Figura 10. Arquitetura geral do SSAR-BR

3.2 Arquitetura

Apesar do objetivo do trabalho ser o desenvolvimento dos módulos de Pré e Pós-Processamento, buscou-se implementar uma arquitetura de *software* suficientemente modular que permitisse a reusabilidade, portabilidade e flexibilidade do sistema como um todo. Seguindo estas premissas, foi desenvolvido um módulo, chamado de Gerenciador, responsável por gerenciar as funcionalidades oferecidas pelo sistema, provendo ao usuário transparência no uso destas funções.

Assim, o módulo Gerenciador torna-se responsável por prover uma Interface Gráfica ao Usuário (GUI) que garante acesso e gerência das funções desenvolvidas, permitindo que uma simulação eletromagnética respeite um *workflow* pré-estabelecido e, por fim, oferece flexibilidade no gerenciamento e integração de outros módulos a serem desenvolvidos.

Pelo fato dos módulos Gerenciador e Modelador Básico compartilharem a mesma GUI, são apresentados nesta seção a descrição e o funcionamento de ambos.

Toda comunicação efetuada entre a *Interface Gráfica do Usuário* e o *Modelador Básico* é feita através do módulo *Gerenciador*, conforme é mostrado na Figura 11. Desta forma é evitado um forte acoplamento entre os dois módulos, o que poderia ocasionar uma dificuldade na manutenibilidade futura do *software*.

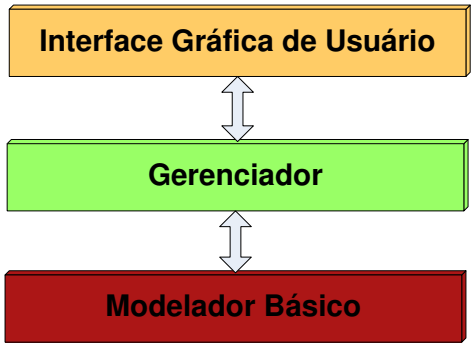


Figura 11. Esquema de camadas do Modelador Básico e Gerenciador

A integração entre as três camadas da Figura acima é mais detalhada no diagrama de classes da Figura 12. Neste têm-se uma visão geral da integração entre as classes que compõem os módulos *Modelador Básico* (classes em vermelho), *Gerenciador* (classes em verde claro) e a *Interface Gráfica de Usuário* (classes em laranja).

O *Gerenciador* tem papel central para o correto funcionamento do *software*, pois além de ser responsável pela comunicação entre o *Modelador Básico* e GUI, também faz a comunicação com os demais módulos do *software* SSAR-BR. Para descrever sua estrutura é feita uma breve explicação do diagrama de classes geral, e, nas seções seguintes, é apresentada suas principais funções com mais detalhes.

Começando pelas classes da camada *Gerenciador*, temos a ***GerenciamentoArvore***, com apenas uma instância de si, contém as principais funções para o gerenciamento da cena de simulação, estas são utilizadas pelas classes das três camadas já citadas (Figura 11) através da intermediação da classe ***GerenciamentoNucleo***.

A classe ***RequisicaoServicos***, localizada no pacote *Mediador*, provê acesso às funcionalidades de outros módulos sem que seja necessário ter uma visão das classes que os compõem. Assim ocorre uma queda no uso de referências entre os objetos do *software* aumentando sua modularidade. Um objeto desta classe é enviado ao ***PainelGerenciador*** no

momento de sua instanciação, em seguida, este objeto (***RequisicaoServicos***) é repassado ao objeto ***GerenciamentoNucleo*** permitindo um controle do acesso a estes serviços.

Já a classe ***ControladorGerenciador***, que implementa a interface ***RequisicaoPainel***, faz o papel inverso da ***RequisicaoServicos***, pois é responsável por disponibilizar o serviços oferecidos pelo módulo *Gerenciador* aos demais módulos do *software*.

As classes ***VerificarStatus***, ***VerificarStatusEvent***, ***Status***, ***VerificarStatusObserver*** e ***VerificarStatusListener*** são responsáveis por implementar o padrão de projeto *Observer* [43]. Este padrão tem por objetivo informar às classes interessadas (observadoras) algum evento ocorrido nas classes observadas. Neste caso, estas classes controlam os seguintes tipos de eventos:

- Evento ***geometriaInserida***: evento gerado pelo usuário quando este adiciona uma geometria na cena de simulação, este ocorre na classe ***Render***. A interface de usuário, representada pela classe ***PainelGerenciador***, é avisada sobre este evento para que esta geometria também seja inserida na "Árvore de Objetos" e repassada às múltiplas vistas (descritas em 3.2.2.1).

- Evento ***fonteInserida***: evento gerado pelo usuário quando este adiciona uma fonte eletromagnética na cena de simulação, da mesma forma que no evento ***geometriaInserida***, a classe ***PainelGerenciador*** é avisada para que esta fonte seja inserida na "Árvore de Objetos" e repassada às múltiplas vistas.

- Evento ***canvasFocus***: evento gerado pelo usuário quando este seleciona, através do mouse, uma das quatro vistas disponibilizadas na cena de simulação. Assim, a classe PainelGerenciador é avisada para que seja possível saber em qual vista o objeto será "desenhado" ou as funções de zoom, rotação e translação serão aplicadas. Além disso, caso o usuário aperte a tecla de espaço, a vista selecionada é maximizada e/ou restaurada.

- Evento ***simulacaoFDTDTerminada:*** evento gerado pela classe ***VerificarStatus*** para informar o status da simulação iniciada pelo usuário. Assim, é gerado um

aviso para uma simulação em andamento, finalizada com sucesso ou que houve algum erro durante o processo.

A classe ***Render*** é responsável pela renderização dos objetos modelados, e com suas quatro instâncias, compõe a cena de simulação formando o elemento múltiplas vistas (Figura 16).

As classes ***PainelGerenciador***, ***JanelaPropriedades***, ***JanelaFEM***, ***JanelaConfigurarGrid*** e ***ImportadorArquivo*** compõem a camada Interface Gráfica do Usuário. Estas são responsáveis pela interação direta com o usuário sem, no entanto, fazerem qualquer alteração na estrutura do *software*. Esta abordagem segue o padrão de projeto MVC (*Model-View-Controller*) [43] no qual tem a premissa de separar os dados da aplicação, definidos como Modelo (*Model*), de sua visão ou interface gráfica (*View*), e as classes que fazem a interação entre a visão e o modelo são definidas no Controlador (*Controller*).

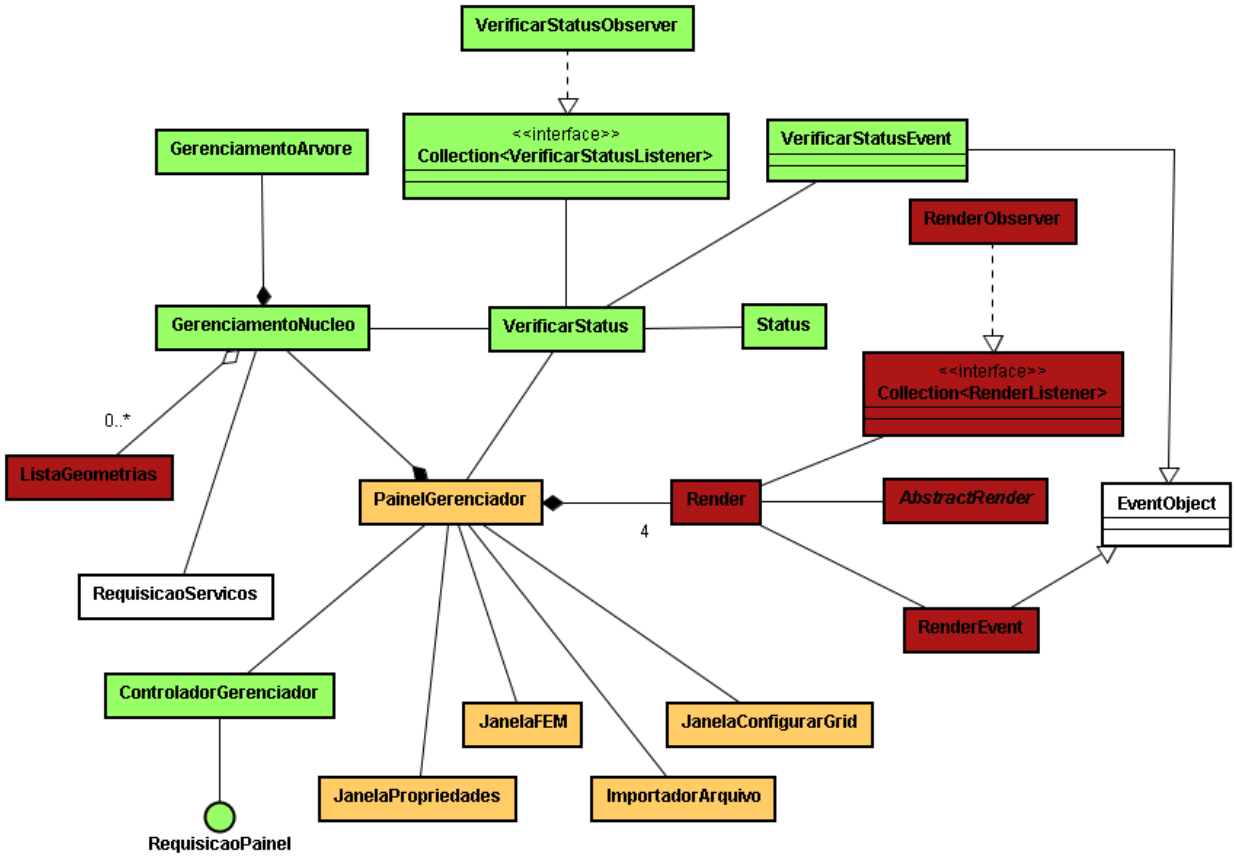


Figura 12. Visão Geral do Módulo Modelador Básico e Gerenciador

3.2.1 *Árvore de Objetos*

A "Árvore de Objetos" é responsável pela visualização de todos os elementos inseridos na cena de simulação, conforme é mostrado na Figura 13.

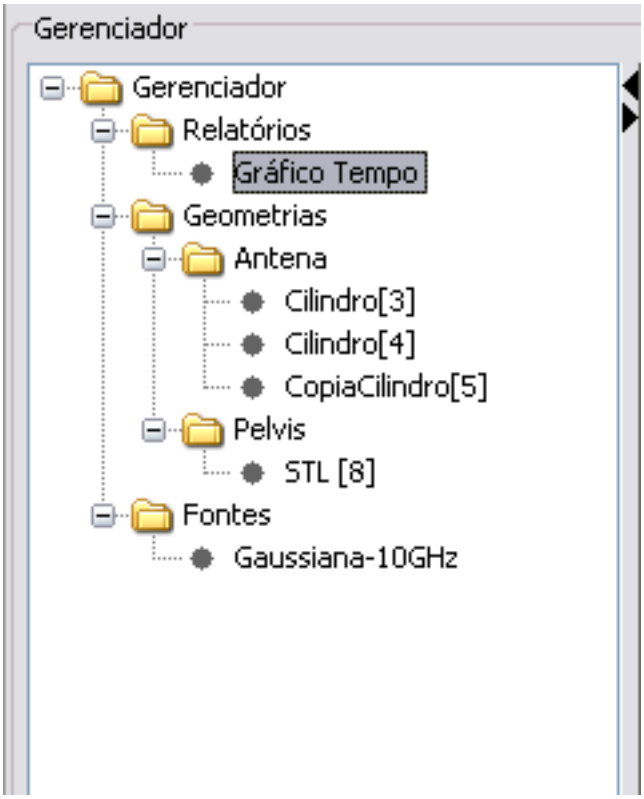


Figura 13. Árvore de Objetos da Cena de Simulação

Esta Árvore possui três categorias em que estes elementos são inseridos, a de Relatórios, Geometrias e Fontes. A categoria Relatórios remete aos gráficos de Pós-Processamento desenvolvidos neste trabalho; assim, após o término de uma simulação, o usuário terá incluso nesta categoria os gráficos previamente escolhidos. As características do módulo de Pós-Processamento serão detalhadas na Seção 3.3 deste trabalho.

A categoria Geometrias contém em uma primeira instância, o objeto ***ListaGeometrias*** que contém uma lista de objetos ***Geometria***. Na Figura acima podemos ver dois objetos ***ListaGeometrias*** que podem representar os elementos "Antena" e "Pélvis" da cena de simulação.

O objeto ***Geometria*** representa modelos geométricos em duas ou três dimensões, conforme Figura 14.

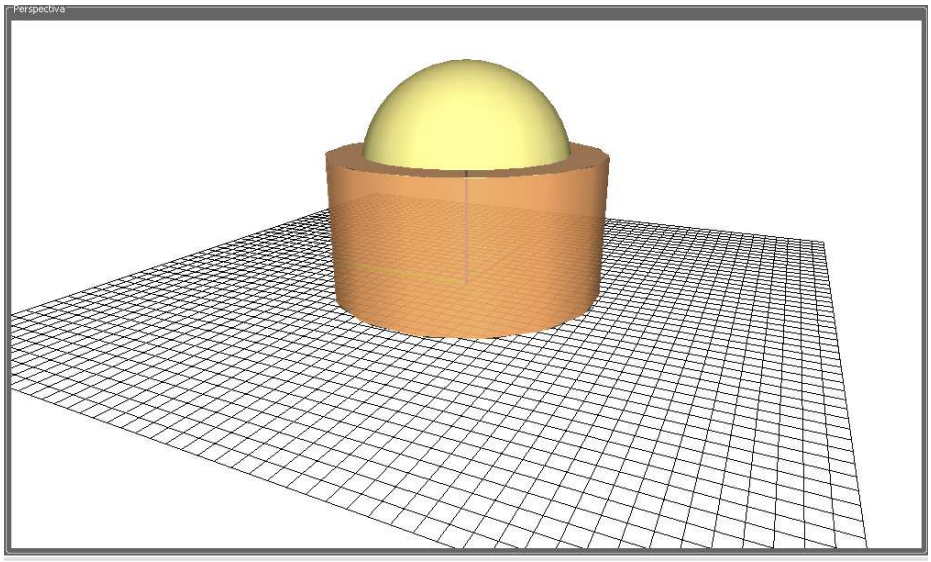


Figura 14. Modelos geométricos representados por objetos Geometria

Estes modelos geométricos podem ser importados de arquivos STL e/ou DXF. Estes arquivos são utilizados em uma vasta gama de *software* CAD 2D/3D existentes no mercado como, por exemplo, AutoCAD©, Maya©, 3D StudioMax©, SEMCAD X©, GiD© e CST EM Studio©.

Por fim têm-se a categoria *Fonte* que contém as fontes eletromagnéticas inseridas pelo usuário, estas representadas por objetos ***FonteEletromagneticaSimulacao***.

Esta estrutura em árvore permite uma facilidade e organização para visualizar, alterar e excluir estes elementos. Por exemplo, caso o usuário tenha um objeto ***ListaGeometrias*** que contenha centenas de Geometrias Básicas (***Geometria***), a cena de simulação provavelmente ficará com desempenho reduzido mesmo que o computador em uso seja de alto desempenho. Desta forma, faz-se necessário deixar-se visível apenas a ***ListaGeometrias*** em que se está trabalhando, e, se for o caso, também é possível selecionar quais Geometrias Básicas serão visualizadas, poupando principalmente o processamento de vídeo.

De acordo com a Figura 15 temos um diagrama de classes que mostra a estrutura da categoria Geometrias. Ou seja, fica claro no diagrama que um objeto da classe ***ListaGeometrias*** pode conter zero ou mais objetos da classe ***Geometri***a, e uma característica implementada durante a fase de testes do *software* é a possibilidade de selecionar uma ***ListaGeometrias*** e escolher um tipo de material para ela, e, após a escolha, todos os objetos ***Geometria*** terão o mesmo tipo de material selecionado, tornando-se um recurso importante para a usabilidade do *software*.

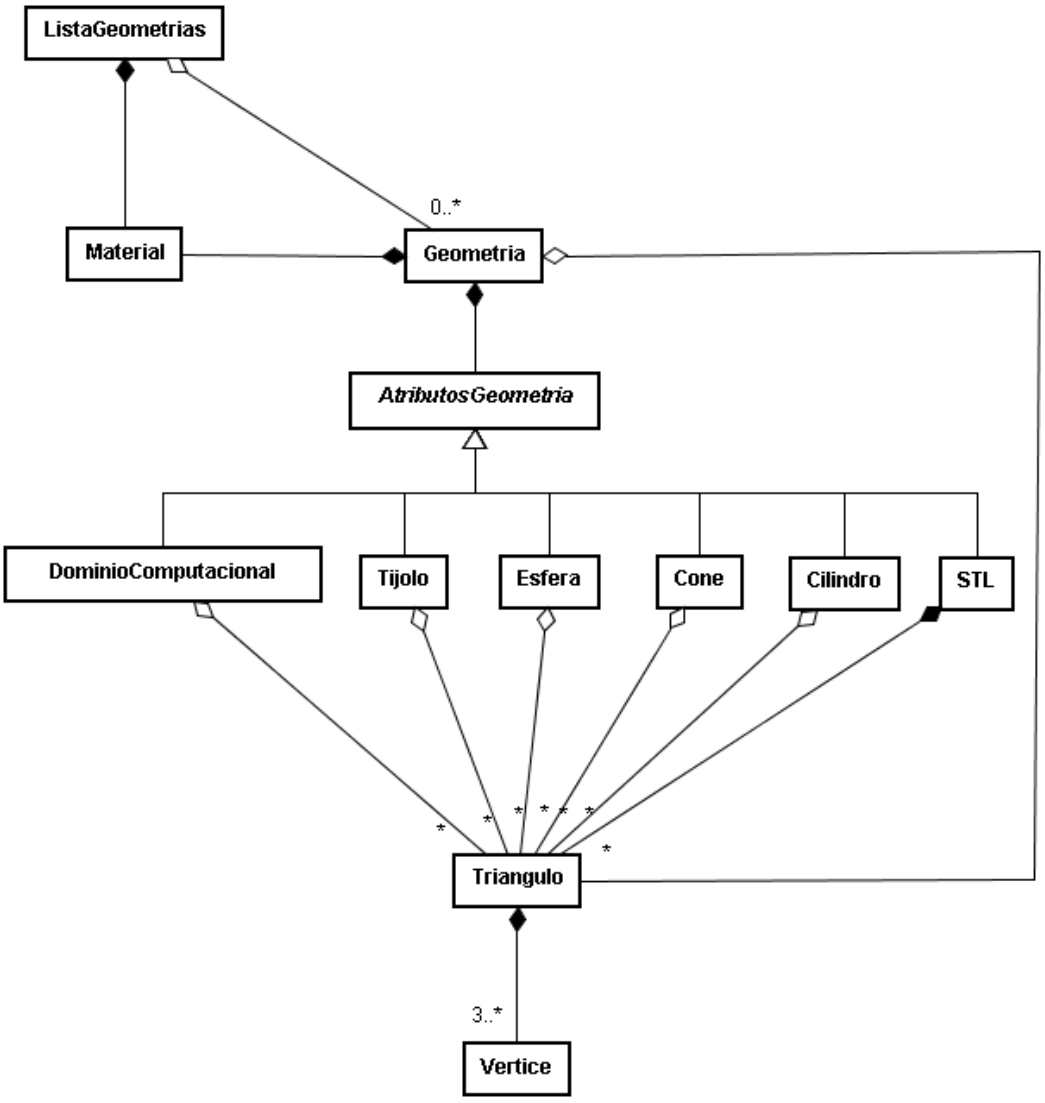


Figura 15. Diagrama de Classes – Estrutura dos Objetos Modelados e/ou Importados

A classe ***Geometria*** é composta por um objeto da classe ***Material***, um da ***AtributosGeometria*** e também por um vetor de objetos da classe ***Triangulo***, este último é composto por três ou mais objetos da classe ***Vertice***.

Uma vantagem desta arquitetura é sua escalabilidade e reusabilidade, pois, caso seja preciso adicionar uma nova forma geométrica (***Geometria***), basta criar uma nova classe filha (concreta) para a classe ***AtributosGeometria*** (abstrata). Um exemplo de escalabilidade é a criação de geometrias irregulares, geralmente modeladas pelo usuário, que ainda não são suportadas por este modelador básico. Se forem implementadas como trabalhos futuros, a arquitetura atual não será modificada, pois basta ao desenvolvedor criar uma nova classe concreta, como por exemplo ***Polígono***, derivada de ***AtributosGeometria***. Outro fator importante é a reutilização da maioria dos métodos já implementados da classe ***Geometria***, principalmente os de rotação, translação e cálculo de seus pontos máximos e mínimos dentro da cena de simulação.

Seguindo este conceito de escalabilidade e reusabilidade, têm-se a classe ***Material***, pois, caso surja a necessidade de criação de novos materiais eletromagnéticos, todas suas características e funções estão encapsuladas nesta classe, assim esta característica não é atrelada à classe ***Geometria***, o que despenderia mais complexidade para uma futura manutenção.

Outra vantagem desta abordagem, é que cada objeto da classe ***Material*** possui uma identificação representada pelo atributo "ID". Esta identificação permite um controle preciso sobre os tipos de materiais eletromagnéticos existentes, independente se estes foram criados pelo usuário ou oriundos da base de dados do *software*.

3.2.2 *Gerenciamento da Árvore*

Algumas funções já citadas como inclusão, exclusão e alteração dos elementos e categorias que constituem a "Árvore de Objetos" são efetuadas na classe ***GerenciamentoArvore***. Esta classe encapsula todas as operações que proporcionam ao usuário acompanhar e operar sobre a cena de simulação.

Segue baixo as operações providas por esta classe:

- ***addFonte***: Adiciona um objeto ***FonteEletromagneticaSimulacao*** à lista de fontes.

- ***addGeometria***: Adiciona um objeto ***Geometria*** que compõe o objeto

- ***ListaGeometrias***. Este objeto é inserido no elemento da árvore previamente selecionado pelo usuário.

- ***addListaGeometria***: Adiciona um objeto ***ListaGeometrias*** à lista de Geometrias da árvore de objetos.

- ***addPreRelatorio***: Adiciona um objeto ***JanelaInserirRelatorios*** na árvore de objetos. Este elemento representa a interface gráfica em que o usuário seleciona os tipos de relatórios de Pós-Processamento.

- ***addRelatorio***: Adiciona um objeto ***Visualizador*** à árvore de objetos. Este objeto representa a interface gráfica dos relatórios de Pós-Processamento descritos na Seção 3.3.

- ***atualizarGeometrias***: Atualiza o objeto ***ListaGeometrias*** inserindo um objeto ***Geometria*** no elemento previamente selecionado.

- ***getListaFontes***: Retorna um objeto **Vector** (da classe ***Java.util.Vector***) que contem objetos ***FonteEletromagneticaSimulacao***.

- ***getListaGeometrias***: Retorna um objeto Vector que contem objetos ListaGeometrias.

- ***getQuantidadeGeometrias***: Retorna o número de objetos ***Geometria*** existentes na árvore de objetos.

- ***getTodasGeometrias***: Retorna um vetor do tipo ***Geometria*** contendo todos objetos ***Geometria*** na árvore de objetos.

- ***getUltimoIdGeometria***: Retorna o ID de valor mais alto do objeto ***Geometria*** inserido na árvore.

- ***getTodosPreRelatorio***: Retorna todos objetos ***JanelaInserirRelatorios*** existentes na árvore de objetos.

- ***getTodosRelatorios***: Retorna todos os objetos Visualizador existentes na árvore de objetos.

- ***getTree***: Retorna o objeto ***JTree***. Este objeto, oriundo da classe ***javax.swing.JTree***, representa a estrutura de árvore que contem as três categorias descritas neste capítulo, *Relatórios*, *Geometrias* e *Fontes*.

- ***remObjSelecionado***: Remove qualquer objeto das três categorias previamente selecionado pelo usuário.

- ***updateTree***: Atualiza a estrutura da árvore no caso de elementos forem removidos ou adicionados.

3.2.3 *Ambiente de Trabalho do Gerenciador / Modelador Básico*

Para demonstrar os recursos dos módulos Gerenciador e Modelador Básico é apresentado seu ambiente de trabalho, visualizado na Figura 16. A interface gráfica é composta por sete itens, o item 1 corresponde a **barra de menus**, logo abaixo segue a **barra de ferramentas** (item 2), à esquerda, abaixo da barra de ferramentas, temos a **árvore de objetos** (item 3), na parte central têm-se as **múltiplas vistas** (item 4), que apresenta a cena de simulação nas perspectivas denominadas “cima”, “lateral”, “frontal” e “perspectiva”. A direita têm-se o painel “**Propriedades**” (item 5), que apresenta as características do objeto selecionado pelo usuário,

logo abaixo têm-se o componente para **visualização dos eixos de coordenadas** (item 6), e por fim, na parte inferior temos a **barra de status** (item 7).

A barra de ferramentas provê acesso às principais funções do *software* SSAR-BR, porém, nem todas correspondem aos módulos desenvolvidos neste trabalho. Os três primeiros ícones, "Novo", "Abrir" e "Salvar" executam as seguintes tarefas, criar um novo projeto, abrir um projeto existente e salvar o projeto atual, respectivamente.

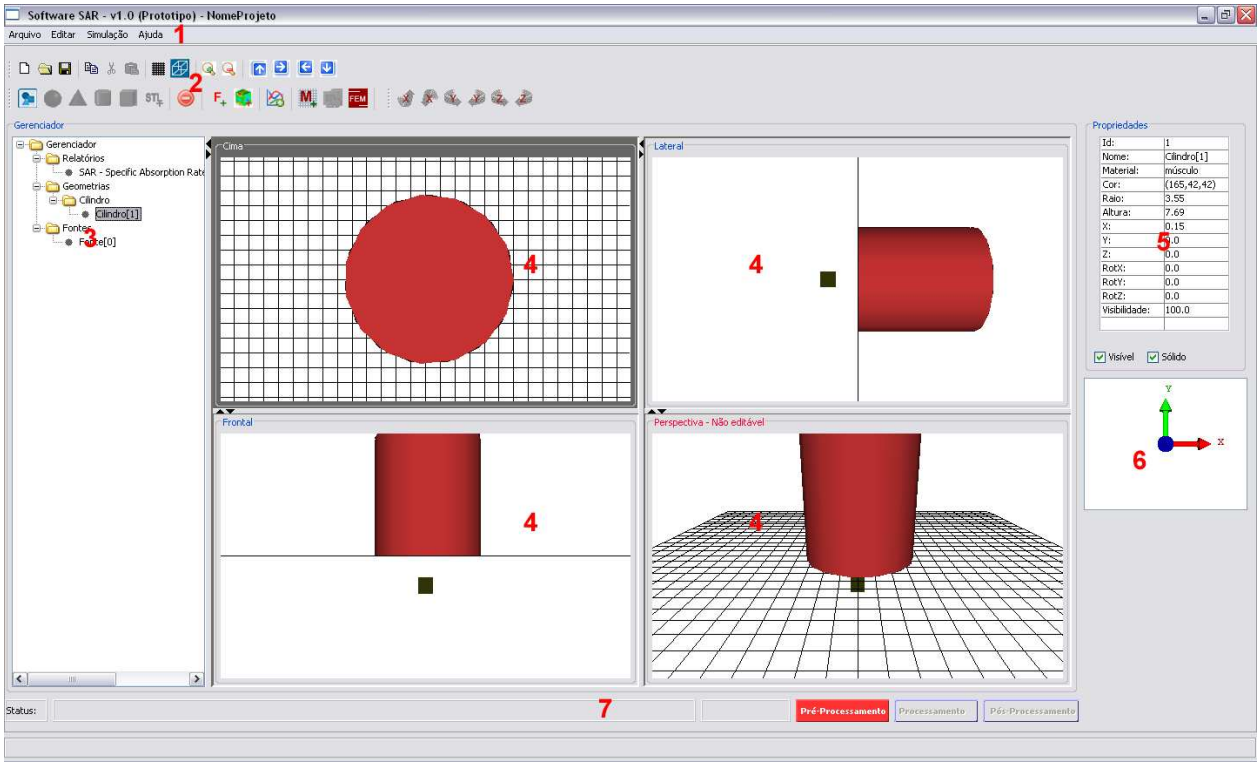


Figura 16. Tela do Gerenciador / Modelador Básico

3.2.4 *Modelagem de Dispositivos*

A modelagem de dispositivos de telecomunicações ou modelos biológicos através de uma ferramenta CAD facilita a montagem da etapa de Pré-Processamento. Em muitas teses de mestrado e doutorado, que envolvem o método FDTD [4-6], têm-se como sugestão de trabalhos futuros o desenvolvimento de uma ferramenta de modelagem. Assim, a especificação de

geometrias feita pelo Modelador Básico traz benefícios para que o usuário possa montar uma cena de simulação de forma rápida e eficiente.

Para validar a ferramenta de modelagem desenvolvida neste trabalho, é utilizado o módulo Gerador de Malha desenvolvido pelo grupo de pesquisa do *software* SSAR-BR [41]. Este algoritmo gerador de malha tridimensional para o método numérico FDTD requer como parâmetro de entrada uma lista que contenha objetos vetoriais. Estes objetos devem satisfazer as seguintes condições:

- Deve ser formado unicamente por faces planas, definidas por três ou mais vértices;
- Nenhuma face do objeto pode cruzar a superfície de outra face do mesmo objeto;
- A superfície do objeto deve ser completamente fechada, isto é, o objeto deve possuir um volume bem definido;
- Cada aresta que compõe as faces do objeto deve ser comum a outra face do objeto, não sendo permitido que uma aresta seja comum a mais de duas faces do objeto.

Desta forma, o Modelador Básico deve construir objetos que satisfaçam estas condições e que também possuem uma estrutura para armazenar o tipo de material que este é composto, definindo as características elétricas e magnéticas do objeto: permissividade, permeabilidade, condutividade e perda magnética, além da densidade específica. Assim, toda estrutura modelada pelo Modelador Básico segue o fluxo visualizado na Figura 17.

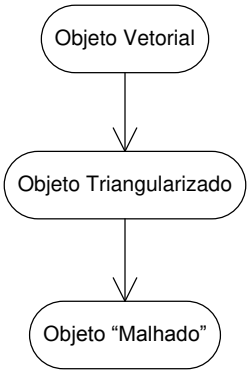


Figura 17. Transição de Objeto Vetorial para Objeto Malhado

Capítulo 4

Pós-Processamento

4.1 Introdução

A visualização gráfica de resultados da camada de processamento é um fator fundamental para realizar-se uma boa análise da simulação. Apesar de existirem no mercado *software* para visualização gráfica como Matlab© e Surfer©, têm-se vantagens em implementar estes recursos integrados diretamente ao *software* SSAR-BR (módulo de pós-processamnto), pois assim são obtidas três grandes vantagens aos usuários.

A primeira está relacionada ao custo de aquisição de licenças, o que não será mais necessário, aumentando assim o valor do *software* SSAR-BR. A segunda trata da usabilidade, pois não é requerido um conhecimento de linguagens como Matlab© e/ou treinamento para geração de gráficos em *software* externos. Neste caso, cabe ao usuário apenas selecionar os gráficos desejados. Por fim é alcançada uma satisfatória reusabilidade do código, o que permite paralelizá-lo em trabalhos futuros, pois, em alguns casos, dependendo do domínio computacional a ser simulado, a geração de animações de duas ou três dimensões requer um alto poder computacional. Na seção seguinte é feita uma descrição de cada relatório e da estrutura geral deste módulo.

4.2 Arquitetura

Conforme é mostrado na arquitetura geral do sistema, Figura 10, seção 3.1, o módulo de Pós-Processamento faz conexão com o módulo Mediador. Esta abordagem trouxe algumas vantagens ao *software* SSAR-BR, principalmente no quesito reusabilidade de código.

Utilizando o conceito de Interfaces da linguagem Java [44], foi definida a interface *RequisicoesPosProcessamento*. Esta é herdada pela outra interface *RequisicaoServicos* que pertence ao pacote *Mediador*. A classe que implementa a interface *RequisicaoServicos* é a *ControladorServicos* também localizada no pacote *Mediador*. Desta forma, é feita uma real integração entre o módulo Pós-Processamento com os demais módulos, pois a classe *ControladorServicos* pode ser usada em qualquer escopo do *software*.

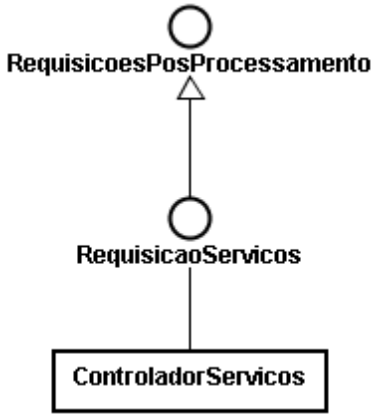


Figura 18. Interface do módulo de Pós-Processamento

Caso surja a necessidade de modificação em algum tipo de relatório do Pós-Processamento basta ao desenvolvedor preservar a assinatura do método definido na classe ***RequisicoesPosProcessamento***.

4.2.1 *Estrutura de Classes*

A estrutura do código desenvolvido pode ser visualizada através do diagrama de classes geral mostrado na Figura 19. Nas seções seguintes é feita uma breve descrição dos tipos de relatórios de Pós-Processamento. Cada tipo de relatório faz uso de um conjunto de classes; por isso, é apresentada uma descrição do funcionamento de cada classe.

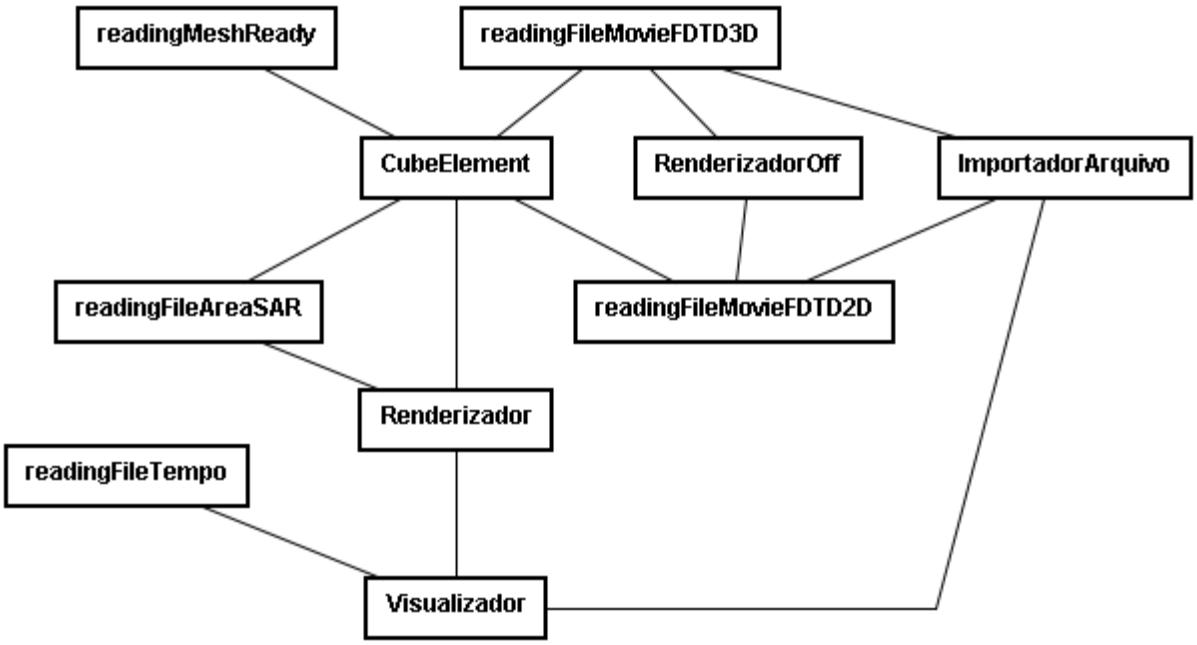


Figura 19. Diagrama de Classes de Alto Nível do Módulo de Pós-Processamento

4.2.1.1 SAR – Specific Absortion Rate – Região do Domínio Computacional

A classe ***readingFileAreaSAR*** é responsável pela leitura dos arquivos gerados pelo módulo de Processamento para visualização do gráfico de SAR em determinada região do domínio computacional. Esta classe recebe como parâmetro uma variável *String* que contém o endereço do arquivo de resultados. Este arquivo contém os valores das coordenadas X, Y e Z e o valor do campo eletromagnético correspondente. Após acessar o arquivo, é feita uma varredura para calcular o maior e menor valor do campo eletromagnético; e retorna-se um vetor que contém objetos da classe ***CubeElement***.

Para se visualizar o uso e interação desta classe na geração do gráfico SAR têm-se o diagrama de seqüência na Figura 20. Deste modo, pode-se ver que a classe ***readingFileAreaSAR*** é responsável apenas pela leitura do arquivo e geração de um vetor de *CubeElement* conforme descrito acima. Em seguida, têm-se a instanciação do objeto da classe ***Visualizador*** que fará a interação direta com a classe ***readingFileAreaSAR*** e, por fim, esta cria um objeto da classe ***Renderizador*** que implementa o código em OpenGL e permite a visualização do relatório (Figura 21).

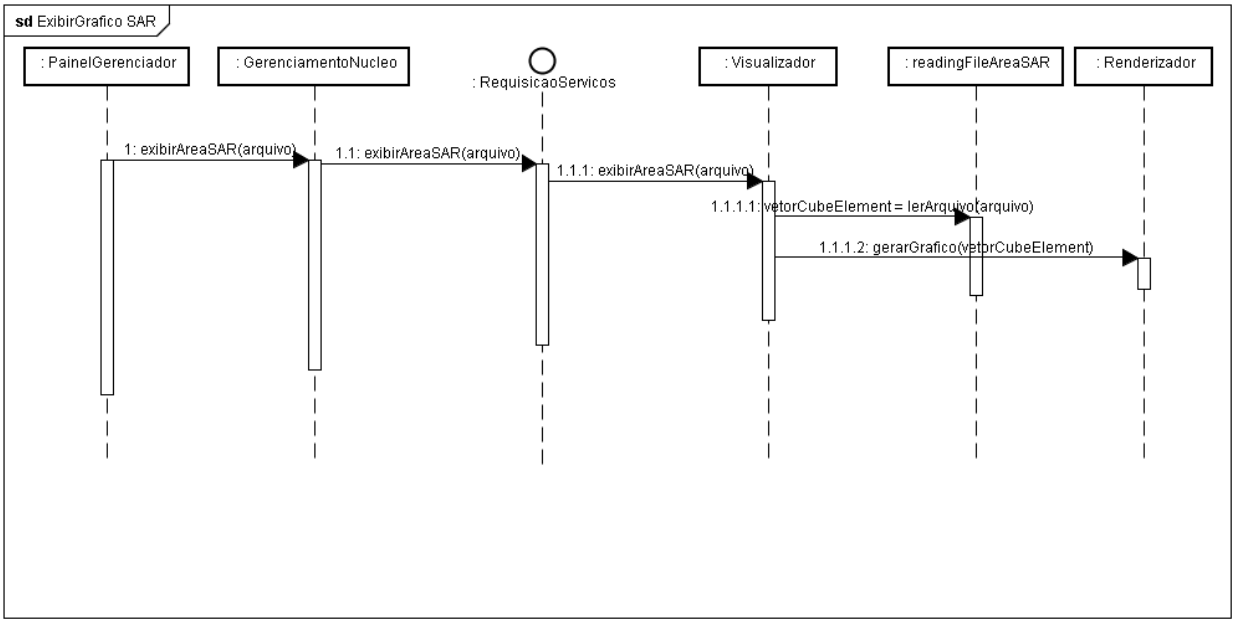


Figura 20. Diagrama de Sequência – Gráfico SAR

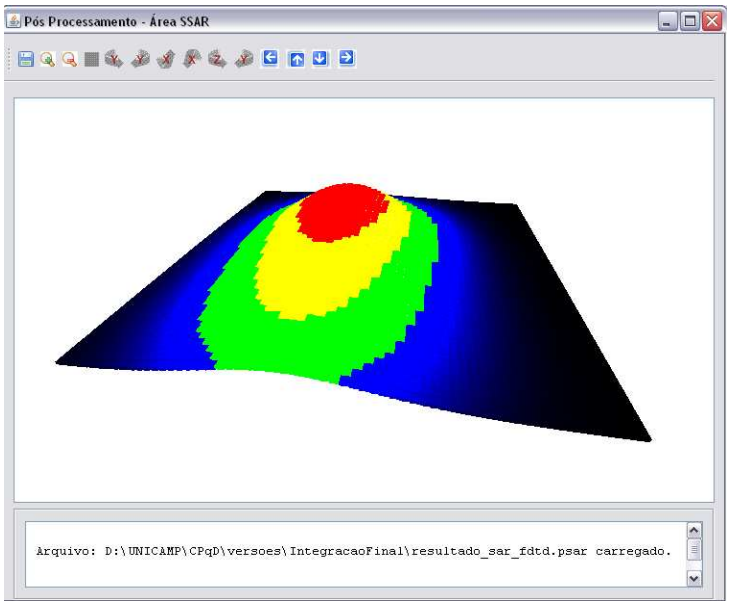


Figura 21. Superfície SAR gerada pelo Pós-Processamento

4.2.1.2 Propagação Eletromagnética no Tempo – Ponto de Referência

A classe ***readingFileFttGraph*** é responsável pela leitura de arquivos gerados pelo módulo de Processamento para o relatório de propagação eletromagnética no tempo como ponto de referência. Esta classe recebe, como parâmetro de entrada, apenas o nome do arquivo, e gera como saída os vetores que armazenam os dados do eixo X e do eixo Y, e também o valor máximo e mínimo do eixo Y.

A interação desta classe com as demais segue o mesmo padrão do gráfico SAR descrito anteriormente com exceção de uma chamada ao objeto da classe ***ControladorGerenciador*** que pode ser visualizada no diagrama de sequência na Figura 20, trocando-se somente o objeto da classe *readingFileAreaSar* por *readingFileTemp*. Esta chamada é feita para que a janela de visualização do gráfico seja inserida na árvore de elementos, o que permite ao usuário abrir esta janela conforme é mostrado na Figura 22.

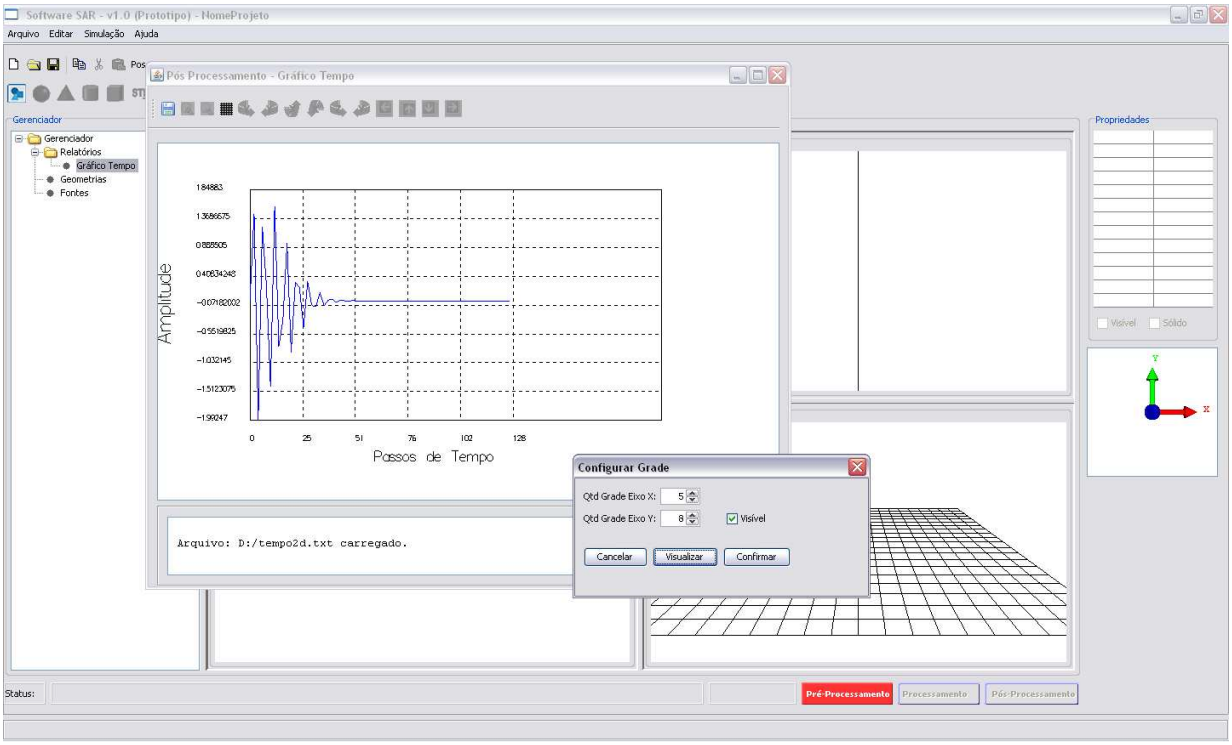


Figura 22. Propagação do Campo Eletromagnético

4.2.1.3 Filme em duas dimensões

A classe ***readingFileMovieFDTD2D*** é responsável pela leitura de arquivos gerados pelo módulo de Processamento para a geração de uma sequência de imagens que formarão um vídeo em duas dimensões da simulação efetuada. O diagrama de sequência visualizado na Figura 23 demonstra que o processo inicia-se da mesma forma que os relatórios anteriores já descritos, porém não há chamada ao objeto da classe ***ControladorGerenciador*** para que este relatório seja inserido na árvore de elementos da interface do *software*. Outra diferença se dá na leitura do arquivo, pois a cada passo de tempo lido, é feita uma chamada ao objeto da classe ***RenderizadorOff*** enviando como parâmetros um vetor de objetos ***CubeElement***, o número do passo de tempo atual e o caminho do arquivo para que a imagem gerada seja salva. Esta etapa é mostrada pelo *loop*, em que cada iteração faz a verificação se o arquivo foi lido por completo.

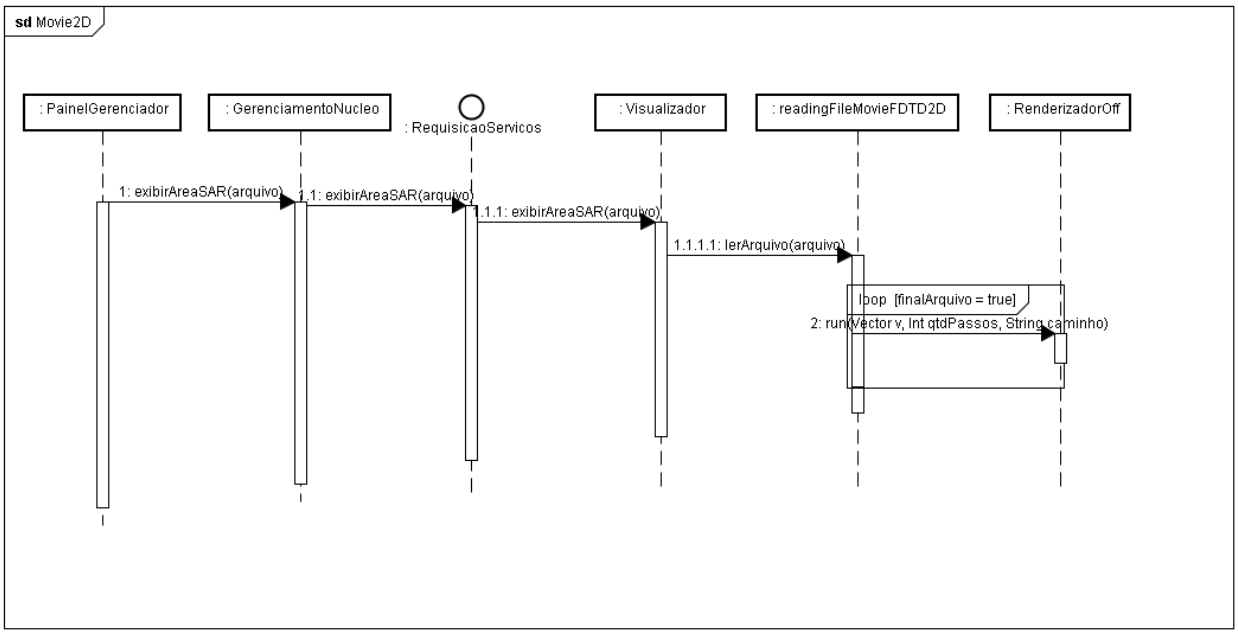


Figura 23. Diagrama de Sequência – Filme 2D

4.2.1.4 Filme 3D

A geração do filme de três dimensões é similar ao de duas dimensões diferenciando-se somente na chamada para a classe ***readingFileMovieFDTD3D***. Esta é responsável pela geração de uma sequência de imagens que formarão um vídeo em três dimensões da simulação efetuada.

4.2.1.5 Visualização de Imagens de Ressonância Magnética

A classe ***readingMeshReady*** é responsável pela leitura dos arquivos de imagens de ressonância magnética já trabalhadas e identificadas em seus diversos tecidos [45] . Estas imagens possuem representações matriciais com resolução de 256x256; um exemplo é mostrado na Figura 24.

Figura 24. Imagem de ressonância magnética tratada e redimensionada em 256x256

Este recurso foi incluído devido à reusabilidade de código, pois devido ao reaproveitamento dos métodos desenvolvidos para a geração de vídeos de duas e três dimensões, foi possível desenvolver um visualizador de malhas previamente geradas a partir das imagens de ressonância magnética (IRM), conforme é mostrado na Figura 25. Como parâmetro de entrada esta classe recebe apenas o nome do arquivo a ser lido, e como saída, retorna um vetor de objetos da classe ***CubeElement***.

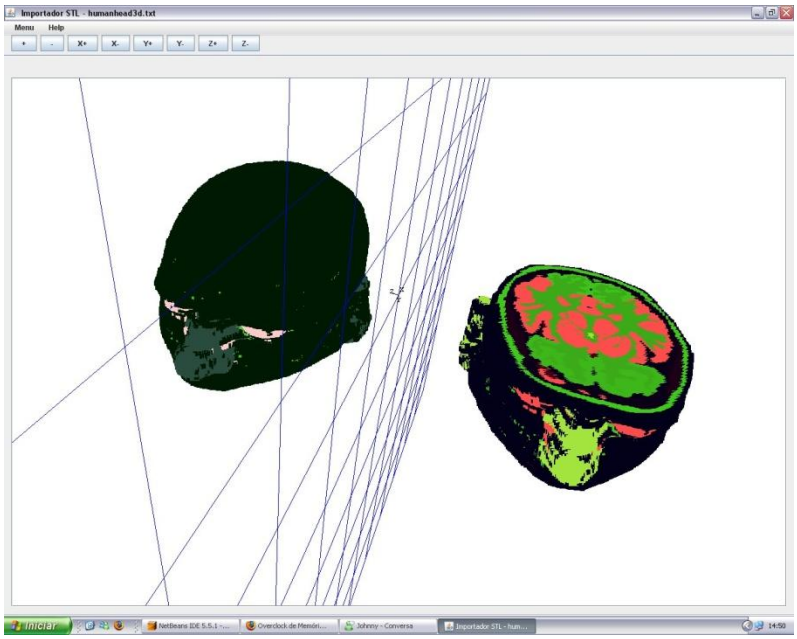


Figura 25. Arquivo de IRM Visualizado no *Software*.

4.2.1.6 Classes de uso comum pelos relatórios

A classe ***Renderizador*** é responsável pela renderização do tipo *OnScreen*, cujo os dados recebidos dos arquivos do módulo de Processamento são visualizados no *display* (monitor) do usuário.

A classe ***GLImage*** é responsável por gerar os *buffers* (espaço em memória) necessários para o processo de renderização *Off Screen* (detalhado no próximo tópico). Ela faz a conversão dos *buffers* em *OpenGL* para o formato que utiliza o padrão de cores RGBA (*Red*, *Green*, *Blue* e *Alpha*). Como parâmetro de entrada ela recebe a largura, altura e o *buffer* que contém os *pixels* da imagem.

A classe ***CubeElement*** é associada às classes responsáveis pela geração dos relatórios Filme 2D, Filme 3D, imagens de ressonância magnética e o relatório de SAR em determinada região do domínio computacional , pois cada objeto representa um elemento gerado tanto pelo módulo de Processamento quanto dos arquivos de imagem de ressonância magnética. Assim, cada objeto contém valores para os eixos X, Y e Z e um valor que pode representar o tipo de

material (imagens de ressonância magnética), onde cada material possui uma cor correspondente, ou então, esse valor representa o campo eletromagnético (elemento das células do FDTD).

A classe ***RenderOff*** é responsável por renderizar as imagens que irão compor o vídeo em duas e três dimensões da simulação do FDTD 3D. Neste caso, cada passo de tempo corresponde a uma imagem gerada, e ao final do processo, o conjunto total de imagens geradas possibilita concepção do vídeo em duas e três dimensões. Por este processo consumir muita memória da placa de vídeo, foi necessário implementar uma renderização denominada *OffScreen*. Este tipo de renderização economiza memória de vídeo, pois de forma padrão, os dados a serem visualizados seriam mostrados no *display*, e com este recurso ativado, os dados são renderizados diretamente em arquivos de imagem JPEG. Como parâmetro de entrada esta classe recebe um vetor de objetos da classe ***CubeElement***.

A classe ***Visualizador*** é responsável por prover a interface gráfica ao usuário, possibilitando assim a realização de diversas funções na renderização de duas e três dimensões. Vale ressaltar que esta classe não realiza diretamente as funções na cena de renderização, mas executa a chamada de métodos existentes nas demais classes, responsáveis por cada operação.

A classe ***RequisicoesPosProcessamento*** é responsável pela integração do módulo de Pós-Processamento com o módulo Mediador do *software*. Definida como uma interface em Java, ela provê as quatro assinaturas que representam os tipos de relatórios previamente descritos. A implementação destes métodos ocorre na classe *ControladorServicos*.

Capítulo 5

Validação e Resultados

Este capítulo apresenta dois diferentes testes que validam os cálculos de SAR com o *software* SSAR-BR. Na primeira seção são apresentados resultados dos testes que comparam o SSAR-BR com [48], onde são apresentados resultados experimentais e simulados com FDTD 3D. Porém, o método referenciado utiliza outras condições de contorno tornando o desempenho do SSAR-BR um pouco mais eficiente.

O segundo procedimento de validação realiza um comparativo entre os valores simulados no SSAR-BR e os resultados experimentais realizados pela equipe do CPqD no recipiente elipsóide próprio para esses testes. Comparativo com o que tem formato da cabeça humana não é viável de ser processado sequencialmente com o recurso computacional disponibilizado para este projeto e por isso, não são apresentados nesse trabalho.

5.1 Validação Inicial do FDTD 3D e SAR

Os testes apresentados nesta seção seguem [48], onde há resultados de simulações com FDTD 3D e os valores de medições laboratoriais de uma estrutura controlada. Para isso é utilizado uma tigela de pyrex preenchida por um líquido que corresponde às propriedades eletromagnéticas médias da cabeça humana (Figura 26) e o respectivo modelo computacional (Figura 27).

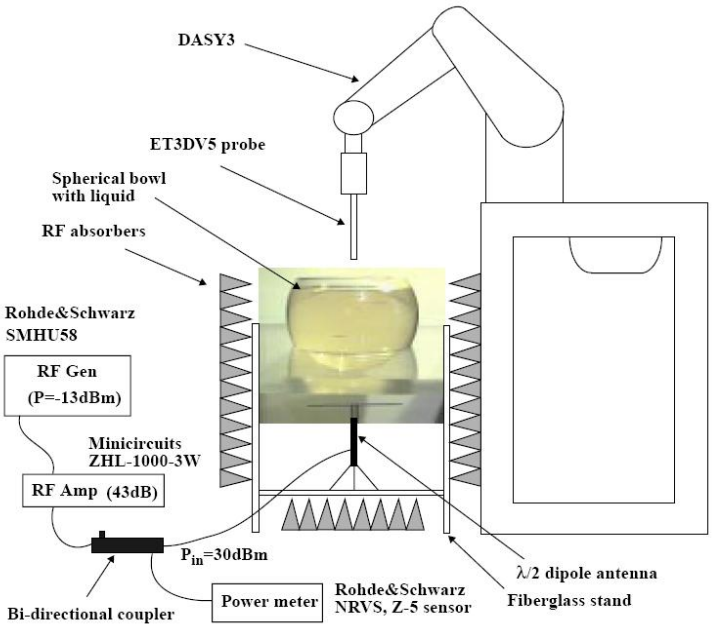


Figura 26. Representação do setup experimental para analisar SAR [48].

A.1 –

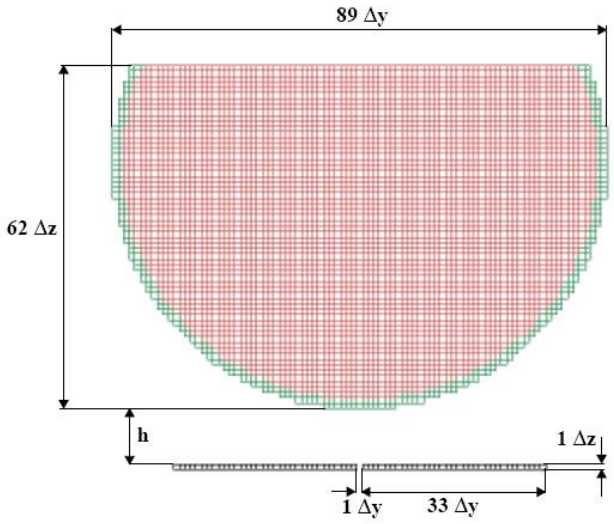


Figura 27. Modelo computacional geométrico do pyrex discretizado em um seção transversal para o FDTD 3D. Considerou-se o domínio computacional total com 165x165x165 células.

O objetivo desse teste é de validar os cálculos de SAR frente a algum resultado apresentado na literatura. Dessa forma, realizamos apenas testes sobre o método numérico FDTD 3D, cálculos de SAR e verificação de que a utilização das malhas geradas está correta.

Nesse contexto, foi gerada uma malha no Matlab© semelhante à desse trabalho [48], ou seja, todas as propriedades de entrada do método foram obtidas através de um programa desenvolvido no Matlab© com vários testes de controle, para atender as especificações de projeto mostrada na Figura 28. Na Figura 29 é mostrado o comportamento da alimentação positiva e negativa do dipolo que foi modelado por uma rampa senoidal.

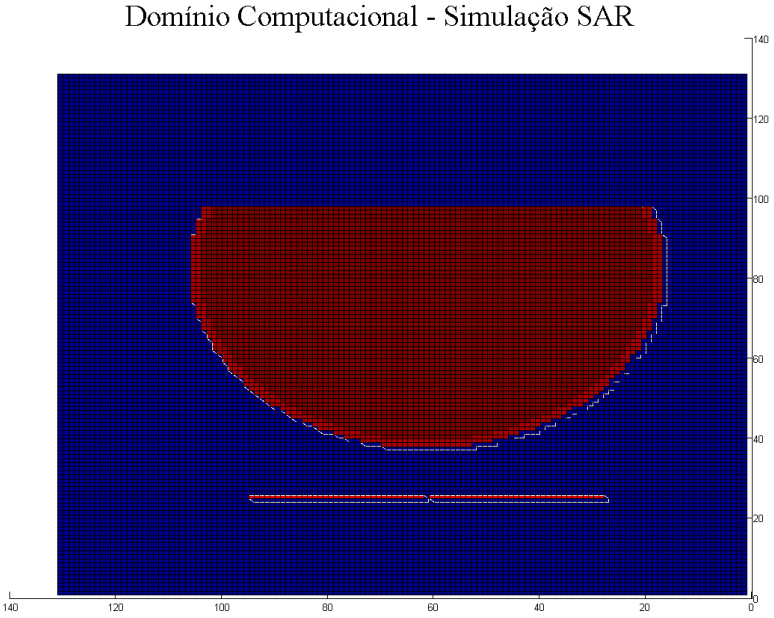


Figura 28. Domínio computacional modelado para os testes SAR.

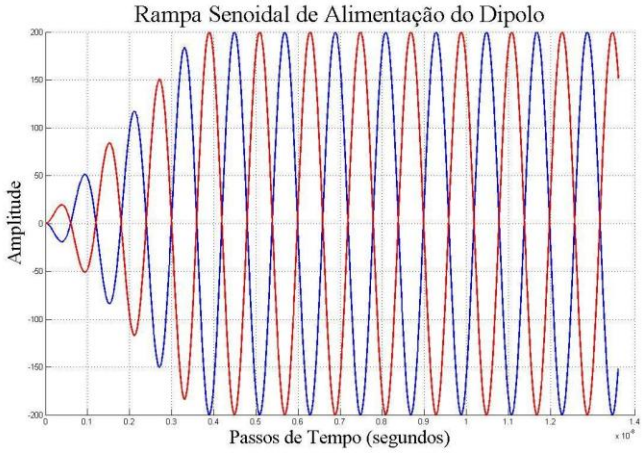


Figura 29. Rampa Senoidal para alimentação do dipolo.

A única avaliação de interesse está em comparar a SAR na região do líquido, como é realizado pelo CPqD. Dessa forma, esses testes iniciais comparam apenas o formato da SAR na região de interesse com o medido e simulado em (Figura 30) [48].

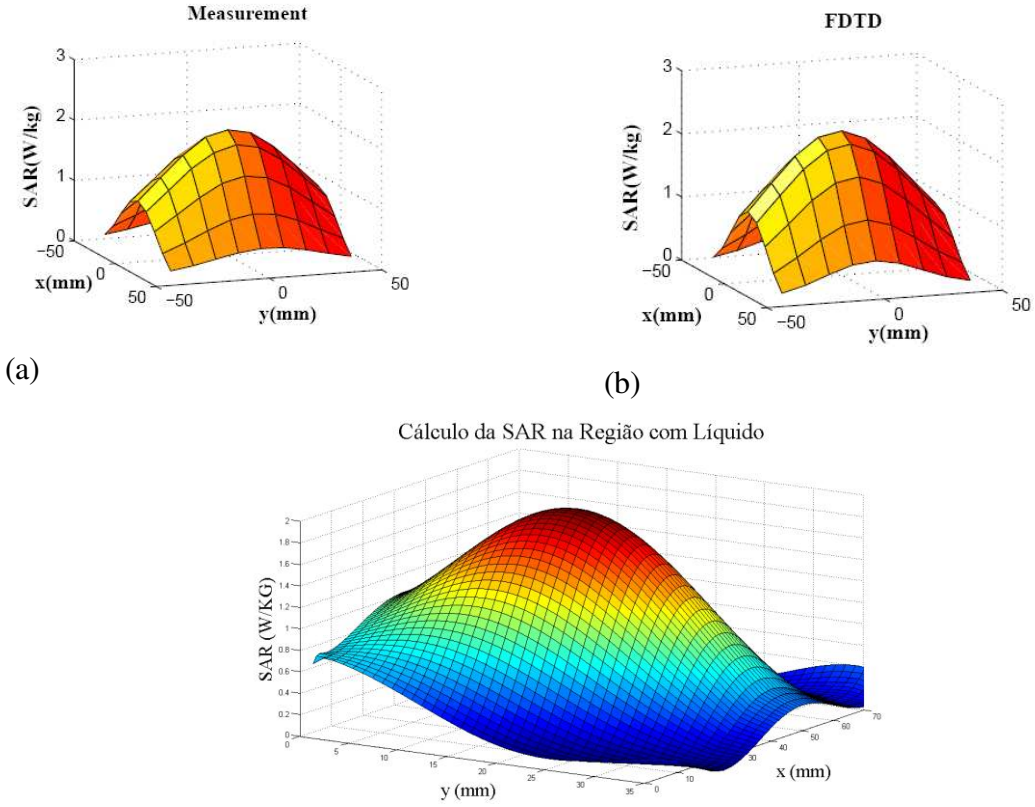


Figura 30. a) Medido [2], b) Simulado FDTD [48] c) Simulado com o FDTD 3D SSAR-BR

5.2 Validação do SSAR-BR com medições feitas no CPqD

O procedimento de validação do SSAR-BR, segundo o projeto inicial, basicamente deve considerar as medições feitas no laboratório de ensaios para radiações não ionizantes do CPqD e comprar com os valores obtidos nas simulações realizadas com o *software*. A variação aceita entre esses resultados de SAR é de 25%. Os procedimentos de medição experimental e as simulações seguem as especificações do padrão IEEE 1528 [49].

Para mostrar os resultados comparativos e a validação desse *software*, previamente é apresentada a sequência de passos necessários para a montagem da cena de simulação (Seção 5.2.1). A seguir, na Seção 5.2.2 são apresentados resultados das simulações considerando apenas o líquido, ou seja, não se considerou o material da bacia que envolve esse líquido. Finalmente, na Seção 5.3.3 é feita a simulação final com todos os parâmetros próximos aos utilizados em laboratório, para analisar a taxa de absorção especifica (SAR); esses valores foram passados pela equipe de medições do laboratório de medições de radiações não-ionizantes do CPqD.

5.2.1 Montando a Cena de Simulação

A primeira etapa para a validação é criar um novo projeto de simulação no *software*. Na tela inicial (Figura 31) o usuário pode selecionar se realmente deseja criar um novo projeto ou continuar a partir de algum projeto já existente.

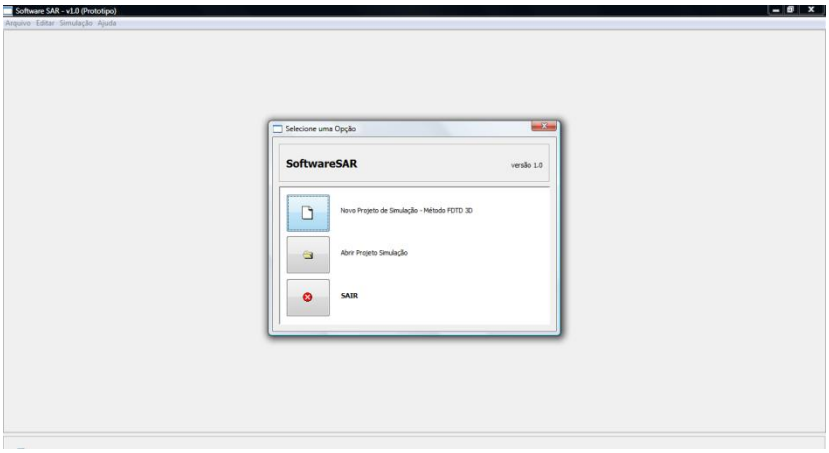


Figura 31. Tela de abertura do SSAR-BR onde o usuário seleciona se deseja criar novo projeto ou abrir algum existente.

Para essa especificação foi iniciado um novo projeto; assim, a janela padrão para novos projetos no *software* é apresentada na Figura 32.

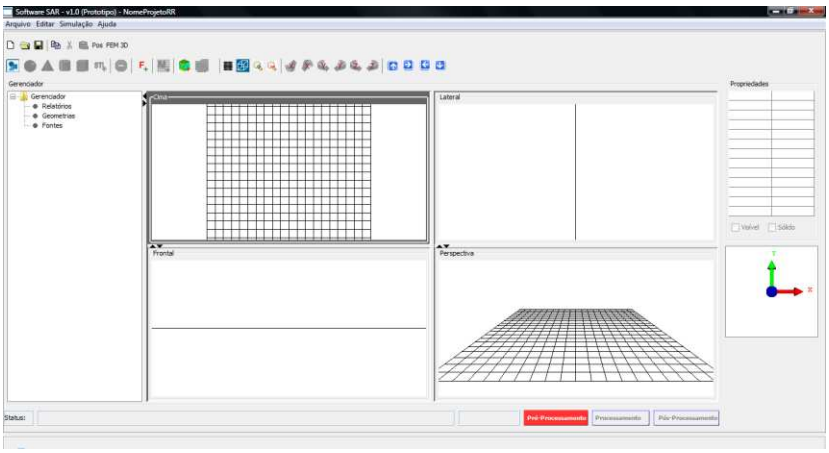


Figura 32. Tela inicial padrão do SSAR-BR para novos projetos.

O projeto do SSAR-BR consta do desenvolvimento de ferramentas para a modelagem de geometrias consideradas básicas (cubo, esfera, cilindro e cone) e a importação de modelos geométricos desenhados em outras ferramentas CAD.

No laboratório do CPqD de medições das radiações não ionizantes existem dois diferentes recipientes para os testes. O primeiro é um modelo humano representado na IEEE 1528 e o segundo é uma bacia elipsóide. O segundo recipiente, a bacia elipsóide, é o que foi utilizado para os testes. Essa opção fundamenta-se na geometria ser menos complexa, passível de ser modelada e processada no FDTD 3D sequencial.

O modelo humano disponível para testes não é viável de ser simulado neste projeto, que contempla apenas o processamento sequencial atual e para resolver esse problema, neste momento, seria necessário processamento paralelo.

A equipe do CPqD modelou esse recipiente no *software* 3D Studio® e exportou para o formato STL. O SSAR-BR possui um pacote para importação de arquivos neste formato. Assim, o passo seguinte foi importar esse modelo (Figura 33).

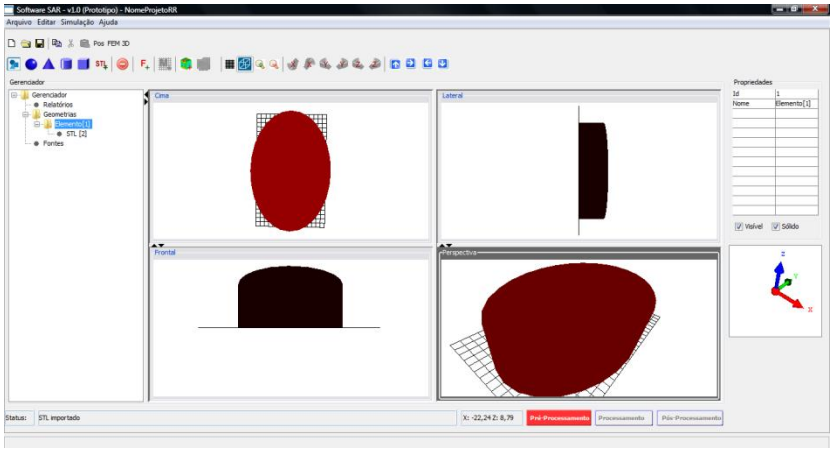


Figura 33. Modelo do recipiente de testes SAR modelado no 3D Studio® e importado para o SSAR-BR.

Para importar esse modelo o usuário precisa criar uma Geometria na barra Comandos e selecioná-la na árvore de Geometrias. A seguir, na barra de Comando pressionar o botão  , que abrirá uma janela para selecionar o arquivo STL de interesse.

Após inserir o modelo ou geometria, o usuário deve informar as propriedades eletromagnéticas destes, ou seja, as características do material (Figura 34). Na primeira simulação de validação do SSAR-BR há apenas materiais a serem considerados, com o líquido que contém as propriedades eletromagnéticas médias dos tecidos humanos e o das barras do dipolo (Cobre) posicionado logo abaixo da bacia.

Caso o material de interesse não esteja cadastrado no sistema, pode-se adicionar um novo material ao selecionar no menu principal a opção Editar, seguida de Cadastrar Material.

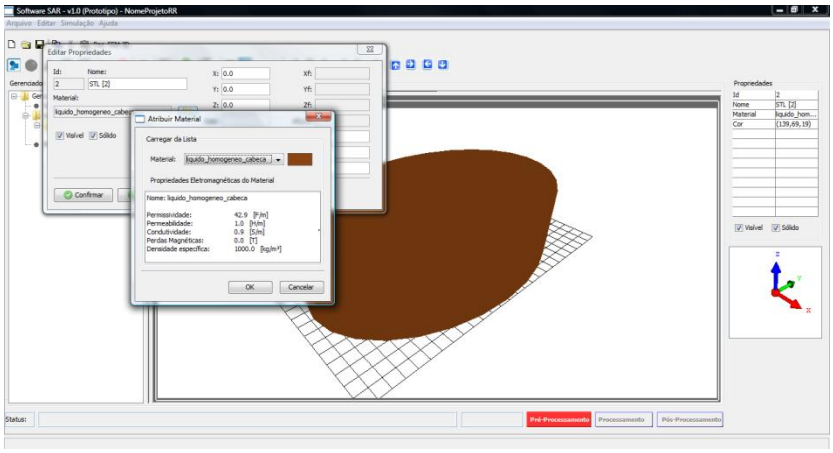


Figura 34. Especificar o material da geometria inserida ou importada.

O passo seguinte é informar a fonte eletromagnética a ser utilizada na simulação. Nesse caso o usuário deve cadastrar uma fonte (Figura 35) com as características de interesse do usuário ou aproveitar alguma previamente cadastrada (Figura 36).

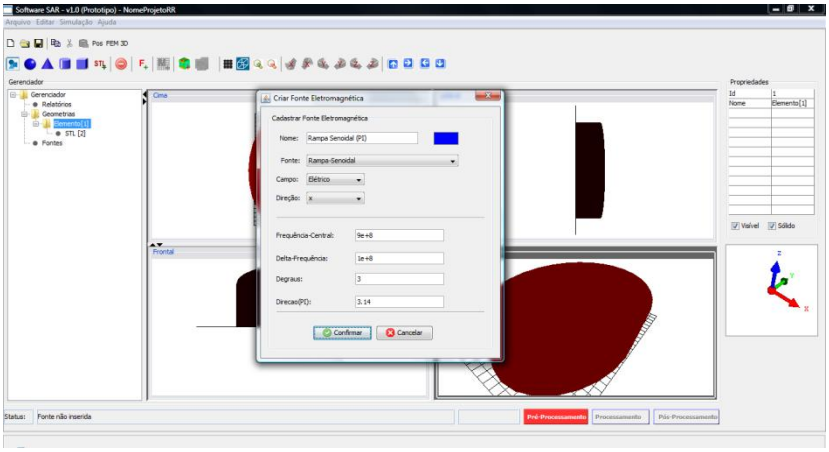


Figura 35. Cadastrar nova fonte eletromagnética para simulações.

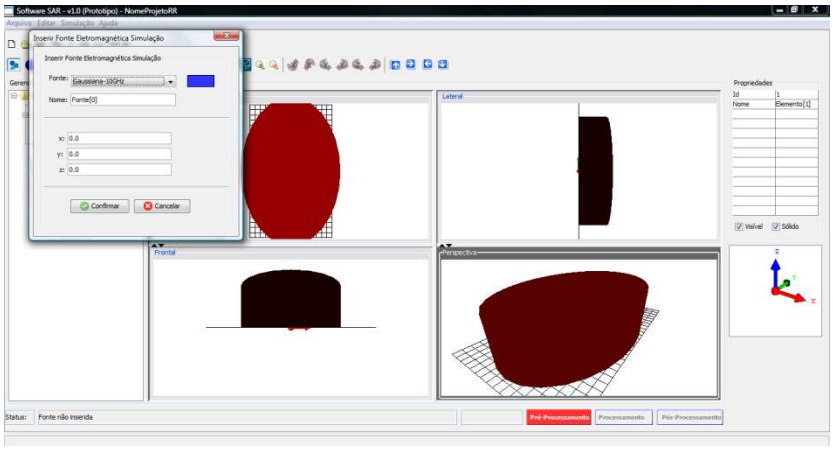


Figura 36. Inserindo fonte eletromagnética cadastrada na cena de simulação.

Após esse procedimento, é necessário apenas informar os relatórios desejados e gerar a malha do domínio computacional a ser simulado (Figura 37).

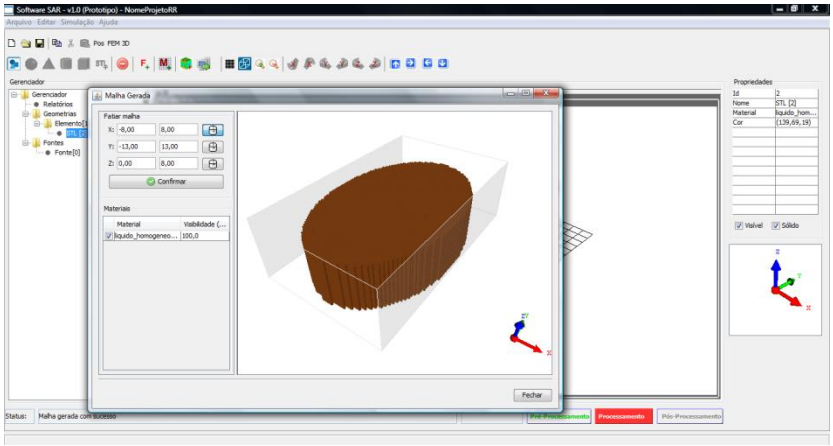


Figura 37. Malha do recipiente elipsóide modelado pela equipe do CPqD**.**

Com todos esses passos devidamente delineados e executados, a cena de simulação de interesse da validação do *software* SSAR-BR estará correta, e o usuário deve pressionar o botão simular para iniciar automaticamente o processamento dos dados pelo método FDTD 3D.

5.2.2 Teste de SAR sem o recipiente

Para essa simulação foi adotado o procedimento anteriormente mencionado para a montagem da cena de simulação. A fonte eletromagnética é a mesma apresentada na Figura A.4. Com a validação dos cálculos de SAR com FDTD 3D no teste anteriormente apresentado, esse teste dá sequência e objetiva validar os recursos da interface do *software* integrados com o método numérico. Desse modo toda a cena de simulação foi projetada no SSAR-BR e a malha gerada automaticamente através dos parâmetros de entrada. Após a inserção dos parâmetros de entrada há o processamento para a realização dos cálculos.

O procedimento de comparação entre os resultados simulados e experimentais foi dividido em dois procedimentos.

No primeiro há uma verificação para identificar a posição do máximo valor de SAR no plano de teste a 2 mm da base do recipiente; essas coordenadas são armazenadas e utilizadas nos planos superiores (Figura 38).

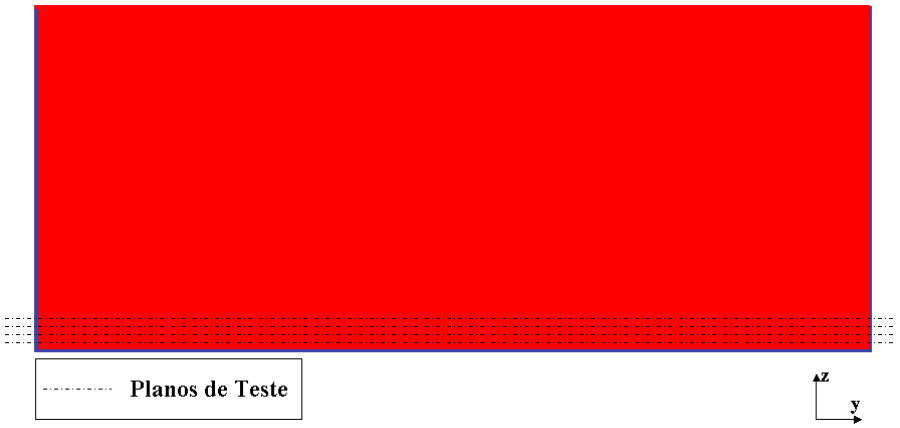


Figura 38. Planos de teste em destaque, da base para o topo um passo de 2mm**..**

Desse modo foram traçados os valores de SAR encontrados nas mesmas posições dessa região para os planos de teste a 2 mm (Figura 39), 4 mm (Figura 40), 6 mm (Figura 41) e 8 mm (Figura 42) da base; e foram comparados.

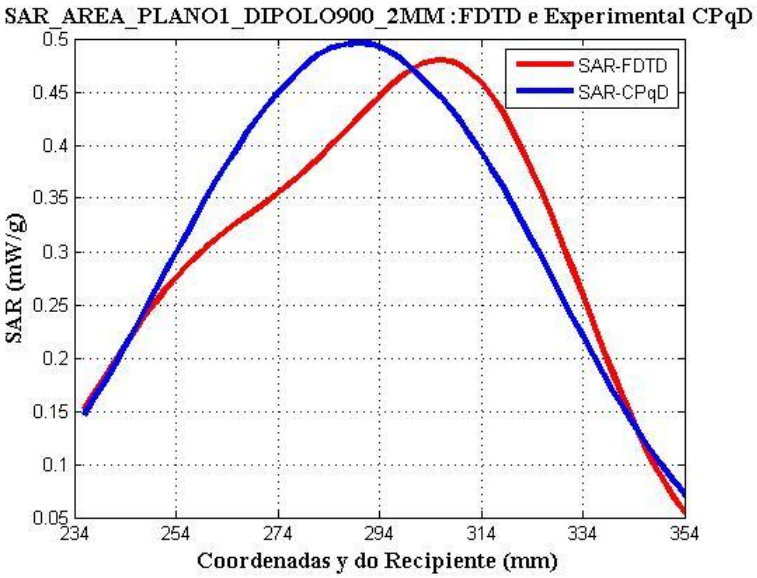


Figura 39. Comparativo do máximo da SAR em simulações sem o recipiente. O plano 1 está a 2 mm da base.

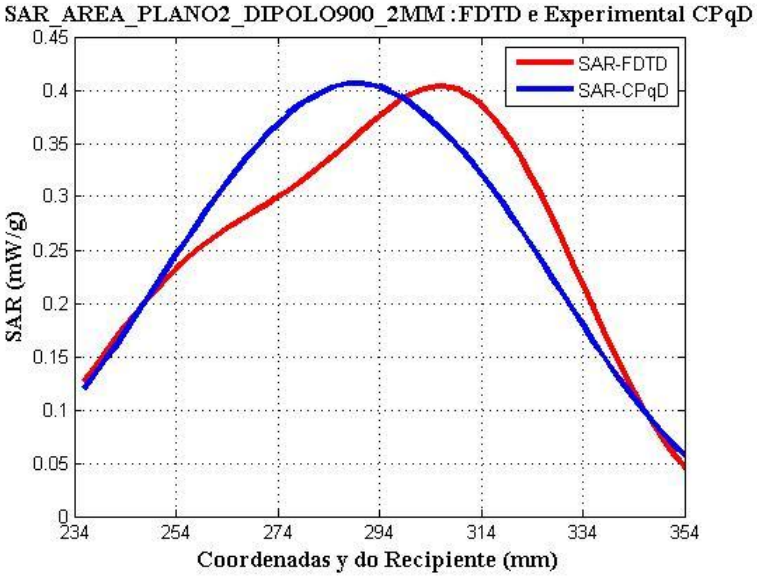


Figura 40. Comparativo do máximo da SAR em simulações sem o recipiente. O plano 2 está a 4 mm da base.

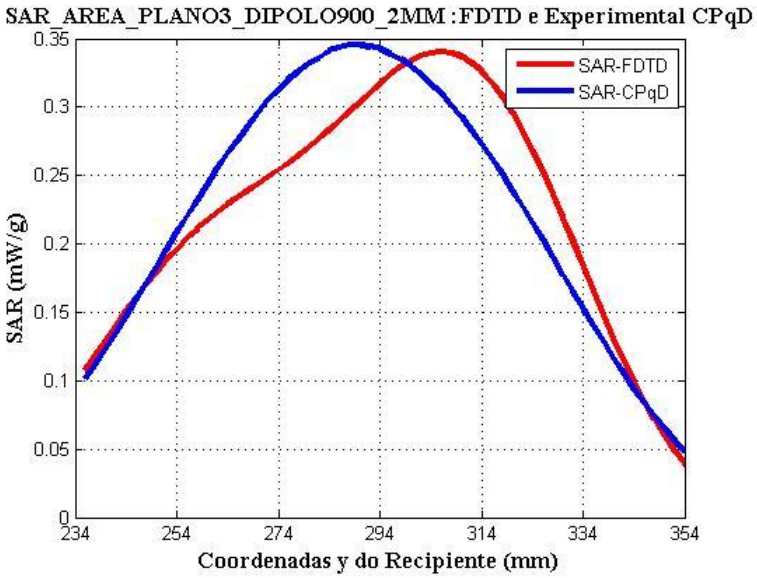


Figura 41. Comparativo do máximo da SAR em simulações sem o recipiente. O plano 3 está 6 mm da base.

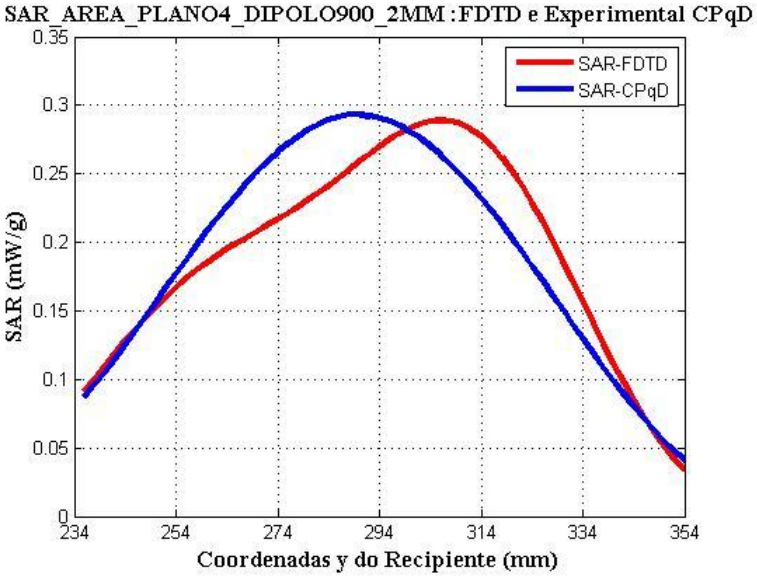
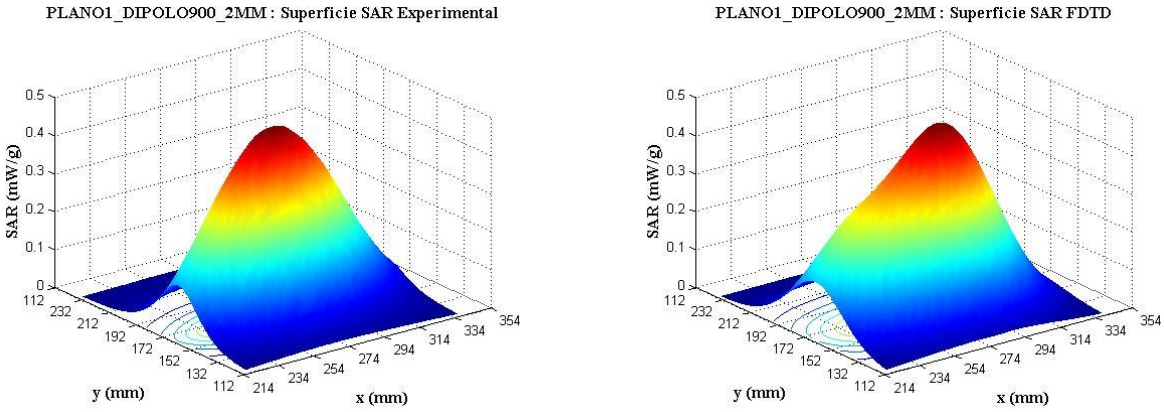


Figura 42. Comparativo do máximo da SAR em simulações sem o recipiente. O plano 4 está 8 mm da base.

A segunda sequência de comparação dos cálculos da SAR apresenta esses mesmos planos de teste, porém agora apresentando toda a superfície desse plano. Isso possibilita que o usuário visualize toda uma região de interesse. Alem disso, demonstra como toda a região apresenta valores próximos e reforçando que a solução convergiu para valores dentro do esperado. Na

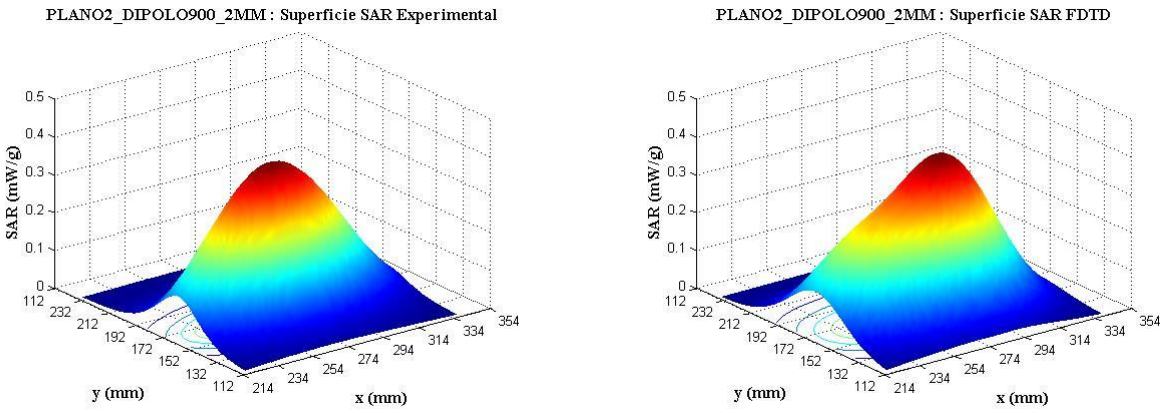
Figura A.18 é apresentado um comparativo dos resultados experimentais (Figura 43a) e os simulados (Figura 43b) entre os valores de SAR na superfície do plano de teste 1.



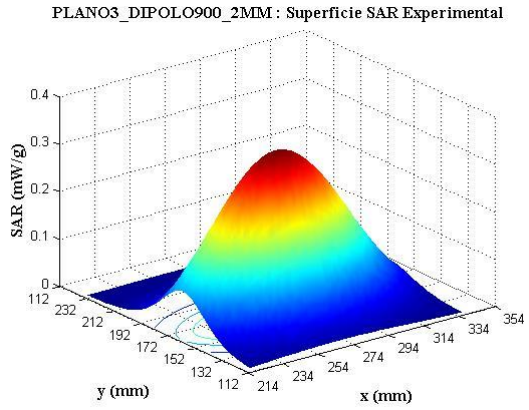
(a) (b)

Figura 43. Comparativo da superfície de SAR no plano 1 (a) superfície experimental e (b) superfície simulada.

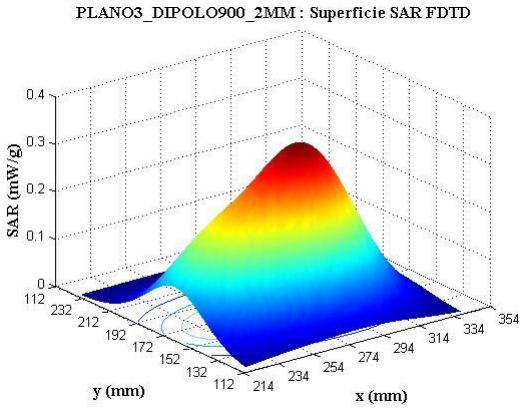
A seguir na Figura 44 são apresentados os comparativos para o plano 2 (Figura 44a e 44b), plano 3 (Figura 44c e 44d) e plano 4 (Figura 44e e 44f).



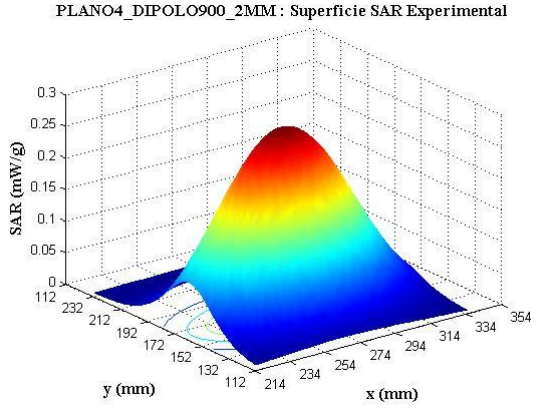
(a) (b)



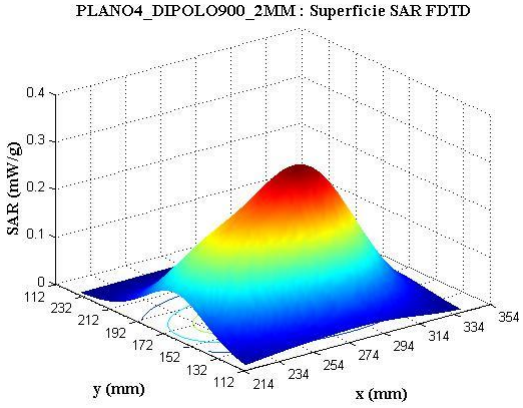
(c)



(d)



(e)



(f)

Figura 44. Comparativo da superfície de SAR no plano 2 (a) superfície experimental e (b) superfície simulada., no plano 3 (c) superfície experimental e d) superfície simulada e no plano 4 (e) superfície experimental e (f) superfície simulada.

5.3 Teste Validação Experimental e Simulador

Para a realização desse teste de validação do SSAR-BR, quanto aos cálculos de SAR, foram necessários alguns cuidados e inserir o recipiente de vidro no modelo que o CPqD enviou em formato STL.

Esse tipo de arquivo STL não possibilita armazenar informações sobre materiais das regiões, apenas mostra as figuras tringularizadas (Figura 45).

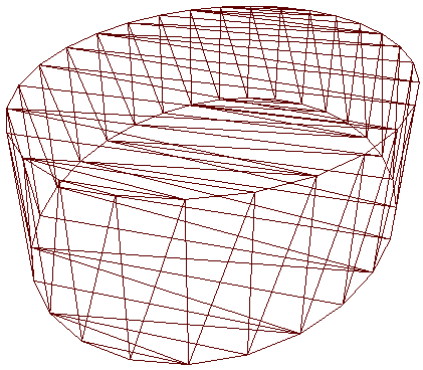


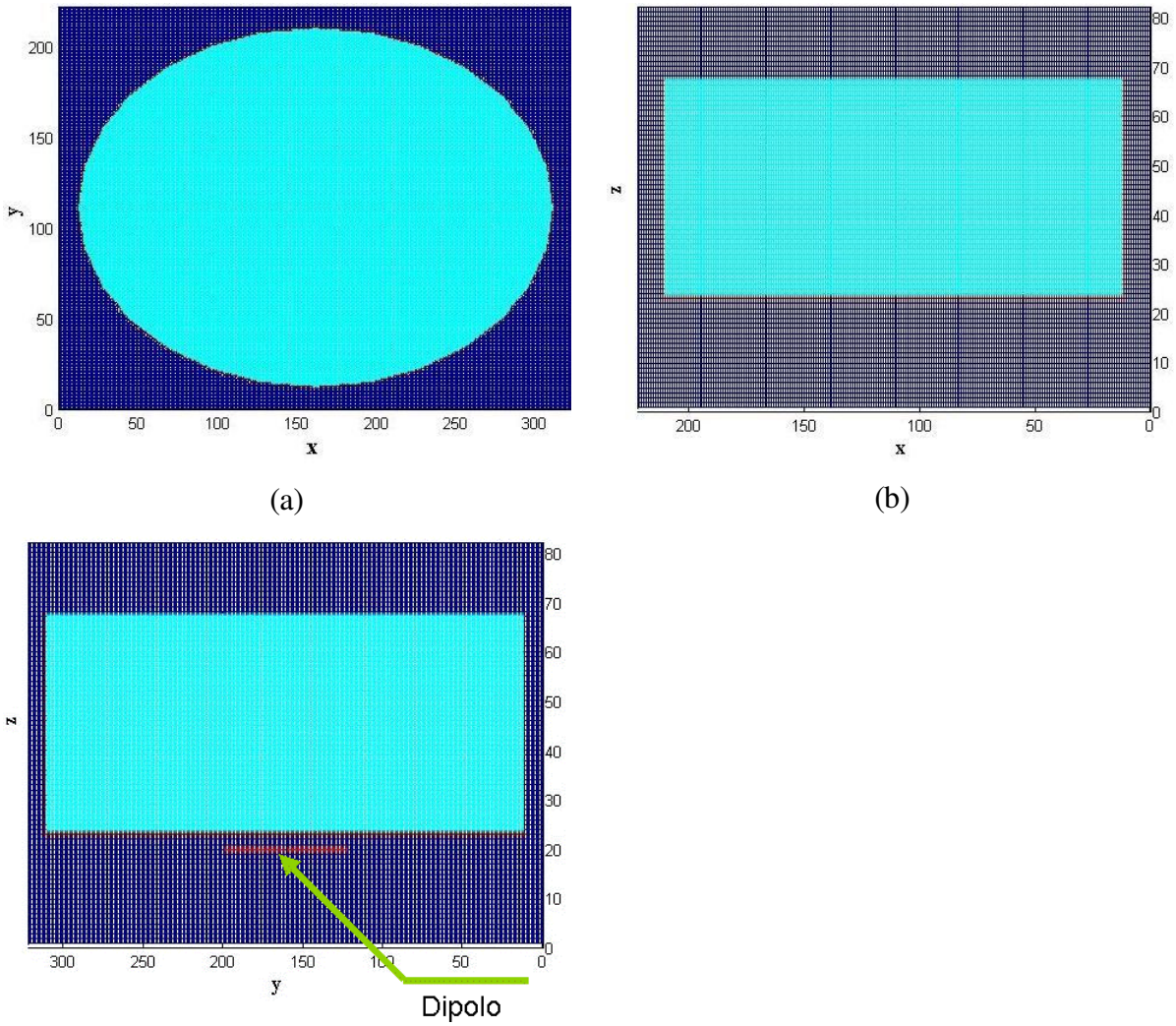
Figura 45. Recipiente de testes no formato STL (triangularizado).

Assim, esse modelo foi importado no *software* e os atributos eletromagnéticos do material foram configurados para a geração da malha. O passo seguinte foi implementar um código Matlab© que possibilitou colocar o vidro na região de interesse do modelo, que é apresentado a seguir.

```
% Aplicativo para inserir o vidro no recipiente de teste
% Material=19 - Vidro   Material=18 - Liquido
clear all;
close all;
matriz = load('br_fdtd_mesh_sem_vidro.isar');    % Arquivo de malha
original
sx = 222;                                        % Quantidade de Elem. em x
sy = 322;                                        % Quantidade de Elem. em y
sz = 82;                                         % Quantidade de Elem. em z
% Criando a matriz de dados para trabalho
cont = 1;
dados = zeros(sx,sy,sz);
for i=1:sx
    for j=1:sy
        for k=1:sz
            dados(i,j,k) = matriz(cont,4);
            cont = cont + 1;
        end
    end
end
```

```
% Gerando a nova malha com o vidro
for i=1:sx
    for j=1:sy
        for k=1:sz
            if (i>1)
                if ((dados(i-1,j,k)==1)&&(dados(i,j,k)==18))
                    dados(i,j,k)=19;
                end
            end
            if (i<sx)
                if ((dados(i+1,j,k)==1)&&(dados(i,j,k)==18))
                    dados(i,j,k)=19;
                end
            end
            if (j>1)
                if ((dados(i,j-1,k)==1)&&(dados(i,j,k)==18))
                    dados(i,j,k)=19;
                end
            end
            if (j<sy)
                if ((dados(i,j+1,k)==1)&&(dados(i,j,k)==18))
                    dados(i,j,k)=19;
                end
            end
            if (k>1)
                if ((dados(i,j,k-1)==1)&&(dados(i,j,k)==18))
                    dados(i,j,k)=19;
                end
            end
        end
    end
end
% Gerando o arquivo com a nova malha que insere o vidro na simula
fid = fopen('br_fdtd_mesh_com_vidro.isar', 'wt');
for i=1:sx
    for j=1:sy
        for k=1:sz
            texto = sprintf ('%d %d %d %d', i-1,j-1,k-1, dados(i,j,k));
            fprintf(fid, '%s\n',texto);
        end;
    end;
end;
fclose(fid);
```

A fonte eletromagnética utilizada nesta simulação é o mesmo dipolo apresentado na Figura 46, estando posicionado no campo E_x e segue a norma IEEE 1528 [1]. Neste caso ela está posicionada a 2 mm abaixo do recipiente como mostrado na Figura 46.



(a) (b)

(c)

Figura 46. Três visões no Matlab© da malha de testes: (a) visão superior do recipiente (b) visão lateral da bacia (c) visão frontal onde se visualiza o dipolo.

O procedimento de testes seguiu o mesmo adotado nos testes em que não foi considerado o recipiente de vidro, ou seja, apenas o líquido de teste com valores eletromagnéticos médios, que seguem a IEEE 1528 [1].

Assim, o primeiro passo é apresentar os comparativos da região de máximos valores SAR (Figura 47).

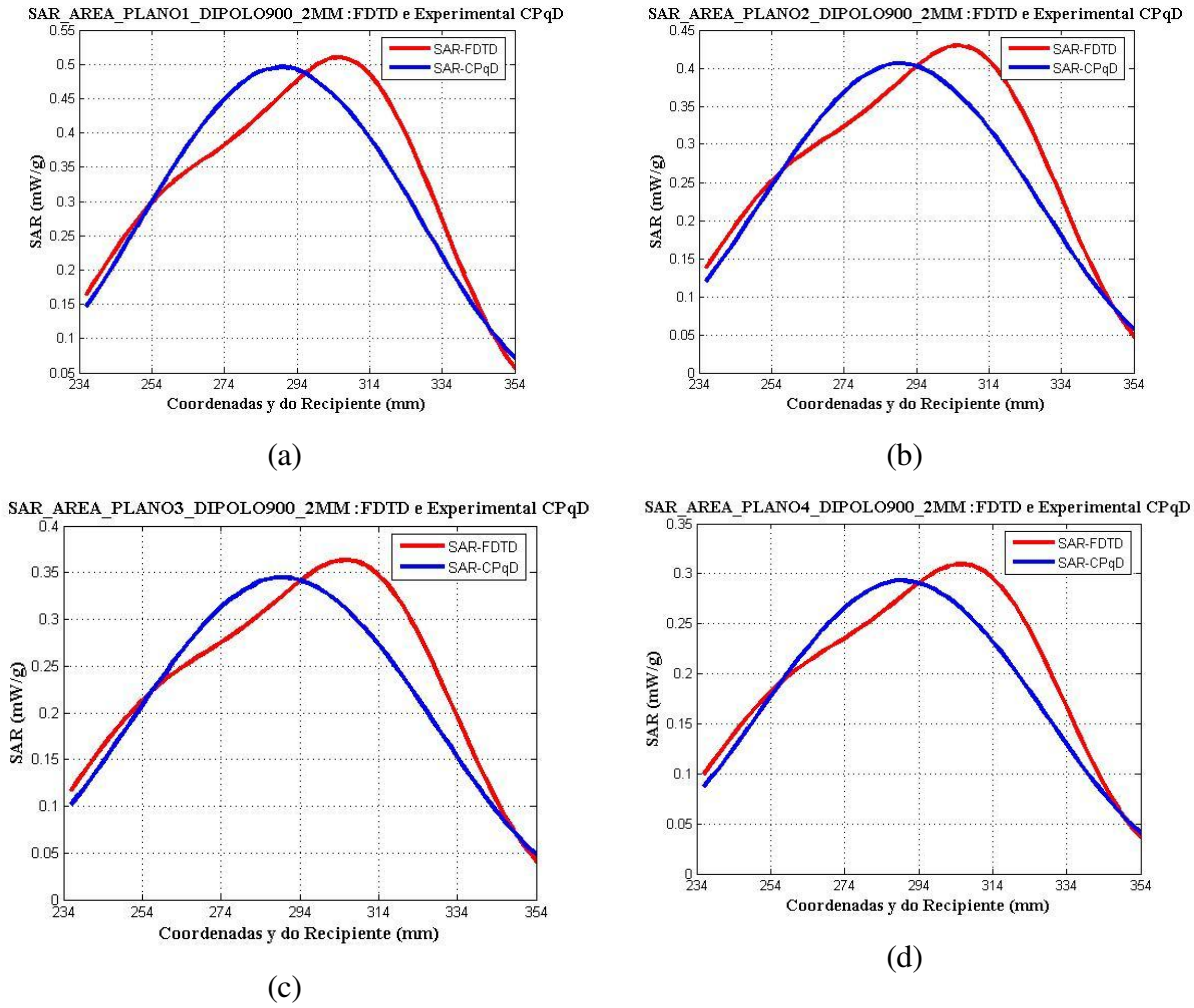
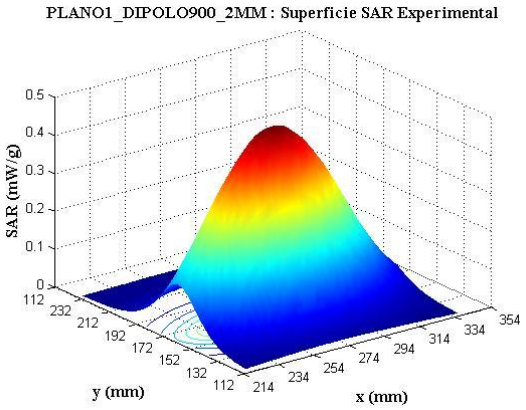
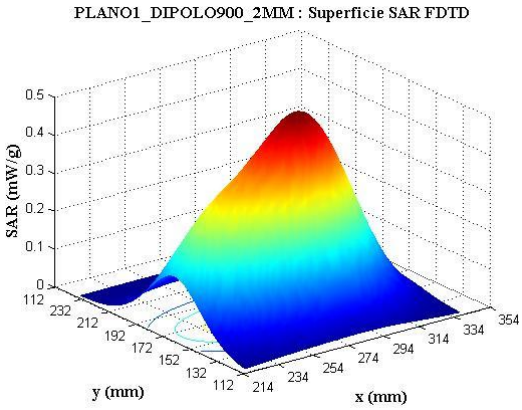


Figura 47. Valores comparativos de SAR considerando o recipiente e o líquido de teste no a) plano1 (b) plano2 (c) plano 3 e (d) plano 4.

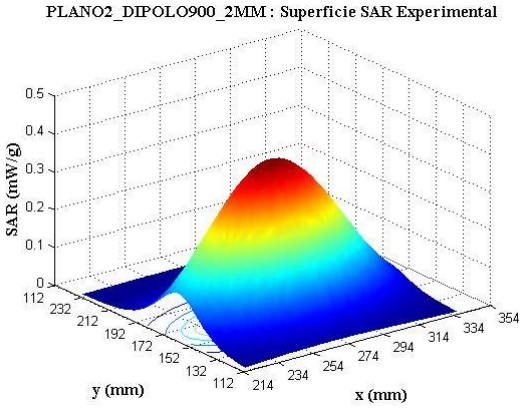
Como também já apresentado no teste sem o recipiente, também são realizadas comparações das superfícies de SAR nos 4 planos (Figura 48).


PLANO1_DIPOLO900_2MM : Superficie SAR Experimental
SAR (mW/g)
y (mm)
x (mm)

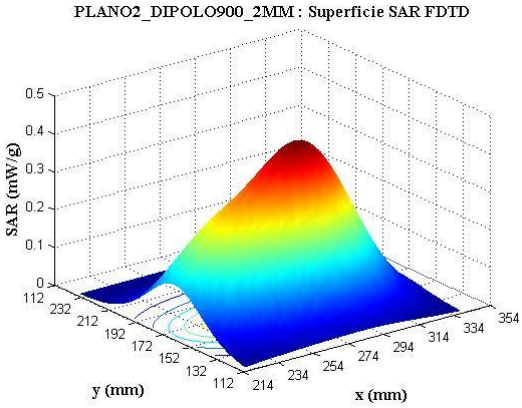
(a)


PLANO1_DIPOLO900_2MM : Superficie SAR FDTD
SAR (mW/g)
y (mm)
x (mm)

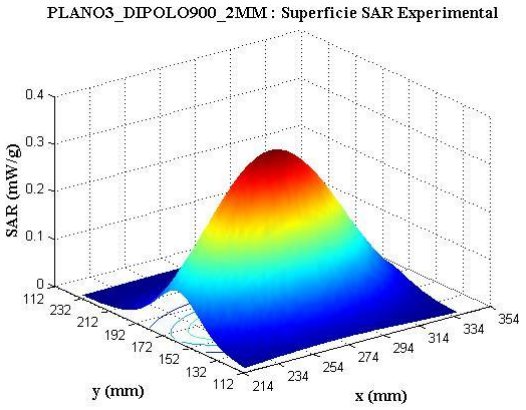
(b)


PLANO2_DIPOLO900_2MM : Superficie SAR Experimental
SAR (mW/g)
y (mm)
x (mm)

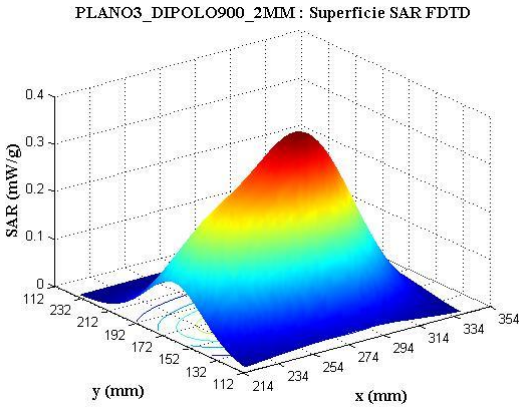
(c)


PLANO2_DIPOLO900_2MM : Superficie SAR FDTD
SAR (mW/g)
y (mm)
x (mm)

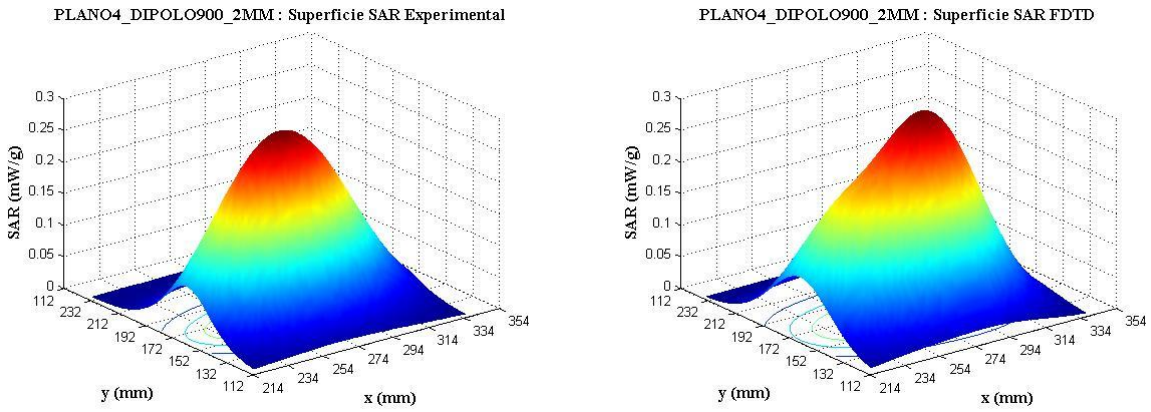
(d)


PLANO3_DIPOLO900_2MM : Superficie SAR Experimental
SAR (mW/g)
y (mm)
x (mm)

(e)


PLANO3_DIPOLO900_2MM : Superficie SAR FDTD
SAR (mW/g)
y (mm)
x (mm)

(f)



(g) (h)

Figura 48. Comparativos das superfícies de SAR experimental e simulado para: (a) plano1 experimental, (b) plano 1 simulado, (c) plano2 experimental, (d) plano 2 simulado, (e) plano 3 experimental, (f) plano 3 simulado, (g) plano 4 experimental e (h) plano 4 simulado.

5.4 Vantagens na utilização do software SSAR-BR

Nesta Seção são apresentadas as principais vantagens, do ponto de vista da usabilidade, do *software* SSAR-BR em relação a dois *softwares* pesquisados na literatura. O primeiro trata-se da comparação com o *software* Meep [50], desenvolvido pelo MIT (*Massachusetts Institute of Technology*), que utiliza o método FDTD na modelagem e simulação eletromagnética; a segunda comparação é feita como *software* desenvolvido em [5], que trata do desenvolvimento e aplicação do método FDTD 3D paralelo.

Como o foco deste trabalho é o desenvolvimento do Pré e Pós-Processamento de simulações eletromagnéticas através do FDTD 2D/3D, as vantagens e comparativos apresentados nesta seção, referem-se exclusivamente ao fator usabilidade, excluindo-se, assim, comparativos de desempenho do núcleo de processamento do FDTD 2D/3D.

5.4.1 *Comparativo com outros softwares*

Em geral, toda simulação eletromangética é composta por sete etapas que podem ser visualizadas através da Figura 49.

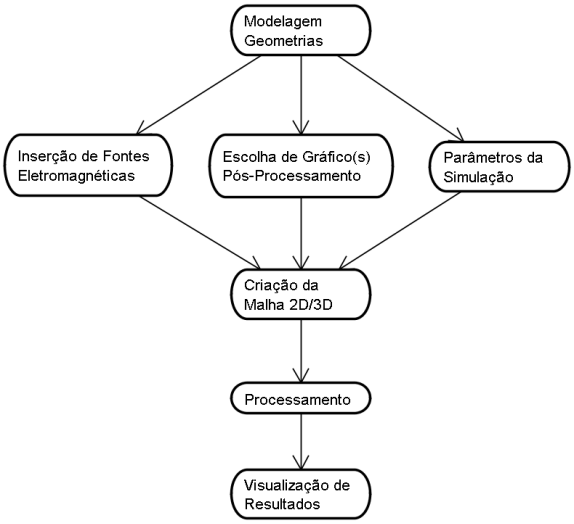


Figura 49. Etapas da simulação eletromagnética

A etapa de modelagem geométrica é considerada bem trabalhosa, pois, além da modelagem vetorial das geometrias, faz-se necessário identificar o material de cada uma. Nos *softwares* sem uma interface gráfica de usuário adequada, todo o trabalho é feito manualmente através da edição de arquivos de entrada, e, em muitos casos, alterações diretas no código fonte.

Para as demais etapas, além das dificuldades descritas acima, há dependências de *softwares* externos dificultando, desta forma, a usabilidade e, consequentemente, forçando o usuário a se aprofundar em conhecimentos fora do escopo da simulação eletromagnética.

Para ilustrar estas dificuldades, temos como exemplo o *software* de simulação que utiliza o método FDTD 3D desenvolvido em [5]; este requer os seguintes passos para que uma simulação seja executada: comandos para chamada do arquivo binário em terminal do sistema, especificação de parâmetros da simulação, modelagem geométrica e inserção de fontes eletromagnéticas (estes dois últimos consistem em alteração no código fonte). Além de todos os parâmetros que devem ser passados para o arquivo executável, o usuário deverá preparar o ambiente computacional, que depende das seguintes bibliotecas e/ou *software*:

- Compilador C/C++
- Instalação do Pacote LAM/MPI para processamento paralelo
- *Software* para visualização de Pós-Processamento (ImageMagik [ref], Matlab[ref] e Octave[ref] e VTK[ref])

Vale ressaltar que o trabalho referenciado não tem como foco o desenvolvimento de uma interface gráfica integrada ao núcleo de processamento, este, na verdade, sugere que trabalhos futuros desenvolvam uma interface gráfica amigável com a finalidade de proporcionar uma modelagem geométrica facilitada, e que, desta forma, o *software* tenha condições de ser reutilizado por outros pesquisadores. Assim, as desvantagens aqui destacadas não se aplicam à dissertação de mestrado utilizada como referência, servindo apenas como teste comparativo.

Outra comparação pode ser feita com o *software* Meep [50] que emprega os seguintes passos e pré-requisitos para montar uma cena de simulação: conhecimento da estrutura do arquivo *ctl*, que significa "*control file*" (arquivo de controle), uso da linguagem *Scheme*, conhecimento da biblioteca *libctl* e, finalmente, do próprio Meep.

Em cada etapa é necessário um estudo da documentação e estrutura do arquivo e/ou linguagem para manipulação dos parâmetros da cena de simulação, fontes eletromagnéticas, geração da malha 2D/3D e visualização dos resultados.

Já no SSAR-BR, toda a interface gráfica de Pré e Pós-Processamento, integradas ao Processamento, provêem ao usuário facilidade em todas as etapas descritas na Figura XX. Esta usabilidade é demonstrada na Seção 5.2.1, e os resultados obtidos também são apresentados no Capítulo 5. Contudo, além da usabilidade alcançada, o objetivo maior da integração entre o Pré, Processamento e Pós-Processamento é o reaproveitamento e continuidade de desenvolvimento do SSAR-BR por outros pesquisadores. Devido a sua arquitetura modular, novos módulos podem ser desenvolvidos e adicionados ao SSAR-BR, além de possibilitar a utilização separada de cada um.

Capítulo 6

Conclusões

Neste trabalho foi apresentado o desenvolvimento dos módulos responsáveis pelos estágios de Pré e Pós-Processamento do pacote de simulação eletromagnética SSAR-BR, e também de um ambiente de computação paralela, baseado no paradigma de Grades Computacionais, que permitirá o uso do mesmo na resolução de problemas eletromagnéticos de grande porte.

No módulo de Pré-Processamento, foi desenvolvida uma ferramenta de modelagem e triangularização de geometrias básicas, proporcionando ao *software* SSAR-BR excelente usabilidade na modelagem de dispositivos a serem simulados, principalmente quando se trata de modelos com alto nível de detalhes, como por exemplo, antenas, terminais móveis e modelos biológicos (humanos e cobaias).

Outro fator importante é a escassez de *software* para modelagem e simulação que se acoplam ao método FDTD, pois na maioria dos casos os *software* disponíveis são comerciais e exigem um grande investimento em aquisição de licenças. Por fim, a estrutura de dados utilizada para representação das geometrias confere uma grande flexibilidade e reusabilidade de código, característica de extrema importância para as atividades de pesquisa realizadas neste Departamento, garantindo assim que outros pesquisadores possam contribuir com melhorias e novas funcionalidades para o *software* SSAR-BR.

No módulo de Pós-Processamento foram desenvolvidos quatro tipos de relatórios subdividindo-se em duas categorias, geração de gráficos e geração de vídeos, ambas servindo de análise para a propagação dos campos eletromagnéticos. Durante seu desenvolvimento, buscaram-se desenvolver resultados similares aos gráficos gerados pelo Matlab© agregando ao *software* SSAR-BR duas vantagens. A primeira se destaca pela usabilidade proporcionada ao usuário, resultando em um aprendizado rápido e evitando que este necessite conhecer qualquer linguagem de programação para a geração dos gráficos. A segunda trata da redução de custos,

pois da mesma forma que ocorre no módulo de Pré-Processamento, o usuário não necessita adquirir licenças de *software* comerciais para visualização dos resultados.

O terceiro componente de *software* desenvolvido neste trabalho trata-se da configuração e implementação do ambiente de Grades Computacionais. Conforme esperado, dois objetivos foram alcançados com este ambiente. O primeiro trata do compartilhamento dos recursos, permitindo maior abrangência das soluções desenvolvidas pelo grupo de pesquisa deste Departamento, pois todos os recursos computacionais existentes poderão ser disponibilizados de forma segura e prática a outros usuários. O encapsulamento destes *software*, como mostrado neste trabalho, não exige conhecimentos avançados do UNICORE 6; assim, a manutenibilidade do ambiente de Grade não fica comprometida, pois outros pesquisadores podem criar novas aplicações e disponibilizá-las de forma ágil. O segundo objetivo alcançado é a implantação de uma infra-estrutura de suporte à programação paralela através do *middleware* UNICORE 6, permitindo assim, uma possibilidade de migração de *software* sequenciais para paralelos. Ainda neste objetivo, esta infra-estrutura provê a opção de realizar simulações através do *software* SSAR-BR em várias máquinas ao mesmo tempo. Apesar deste *software* empregar o método FDTD sequencial, esta abordagem se mostra interessante quando há necessidade de se executar diversas simulações com parâmetros diferentes, ou seja, ao invés de instalar o *software* em várias máquinas, e executá-las manualmente, ou até mesmo rodar uma simulação após a outra, o usuário pode definir um *job* para cada máquina e enviá-lo através da rede local ou internet.

Além das contribuições já descritas, este trabalho permite uma abordagem educacional, onde o *software* SSAR-BR, integrado ao ambiente de Grades Computacionais, pode ser utilizado de duas formas, localmente e remotamente. No uso local, temos o uso das ferramentas de modelagem de dispositivos e visualização das simulações eletromagnéticas. Já no uso remoto, inserem-se os casos onde o domínio computacional a ser simulado seja de grande porte, no qual um computador comum não seria capaz de executar tal simulação ou, na melhor das hipóteses, esta implicaria em muito tempo de processamento, inviabilizando seu uso didático. Desta forma, o ambiente de Grades Computacionais contribui para o envio remoto deste tipo de *job* a algum *Cluster* disponível de forma prática ao usuário, conforme apresentado na Seção 2.5 motivando alunos da área de computação e engenharia elétrica a usarem esta ferramenta.

Por fim, conclui-se que este trabalho colaborou de forma significativa com as atividades do grupo de pesquisa e proporcionou conhecimento sobre Grades Computacionais para que trabalhos futuros explorem todas as vantagens oferecidas por este tipo de ambiente.

6.1 Trabalhos Futuros

A continuidade do desenvolvimento do *software* SSAR-BR é de interesse do grupo de pesquisa, deste modo, têm-se como objetivo desenvolver novas funcionalidades e aperfeiçoar as já existentes. Segue abaixo algumas sugestões de trabalhos futuros:

- Implementação de um gerador de vídeos de duas ou três dimensões em paralelo, pois, dependendo do domínio computacional a ser modelado e a quantidade de passos de tempo utilizado no método FDTD, o Pós-Processamento torna-se inviável computacionalmente ou despende-se muito tempo para geração destes vídeos.

- Incrementar o Pós-Processamento para a criação de outros tipos de gráficos inerentes ao método FDTD.

- Incrementar recursos avançados ao Modelador Básico como, por exemplo, ferramentas de modelagem e triangularização de geometrias irregulares em três dimensões.

- Integrar o ambiente de Grades Computacionais ao *software* SSAR-BR para que seja possível enviar uma cena de simulação automaticamente aos *Clusters* previamente configurados pelo usuário. Estes poderão estar localizados no próprio Departamento ou em outros institutos que utilizem um *middleware* compatível com o UNICORE 6.

Vale ressaltar que a arquitetura baseada em componentes, desenvolvida no *software* SSAR-BR, pode agregar outras aplicações desenvolvidas pelo grupo de pesquisa para que se torne um pacote completo em simulação de campos eletromagnéticos.

Referências Bibliográficas

[1] T. C. Moronte, **Uma infra-estrutura de *software* para apoiar a construção de arquiteturas de *software* baseadas em componentes**, Dissertação de Mestrado, Instituto de Computação (IC), Universidade Estadual de Campinas (UNICAMP), 2007.

[2] J. P. da Silva, **Simulação por Elementos Finitos da Propagação de Feixes Ópticos em Estruturas Fotônicas**, Tese de Doutorado, Departamento de Microonda e Óptica (DMO), Faculdade de Engenharia Elétrica e de Computação (FEEC), Universidade Estadual de Campinas (UNICAMP), 2003.

[3] A. M. F. Frasson, **Simulação, por Elementos Finitos 3D, de Problemas Eletromagnéticos no Tempo e na Frequência**, Tese de Doutorado, Departamento de Microonda e Óptica (DMO), Faculdade de Engenharia Elétrica e de Computação (FEEC), Universidade Estadual de Campinas (UNICAMP), 2002.

[4] C. H. S. Santos, **Computação Paralela Aplicada a Problemas Eletromagnéticos Utilizando o Método FDTD**, Dissertação de Mestrado, Departamento de Microonda e Óptica (DMO), Faculdade de Engenharia Elétrica e de Computação (FEEC), Universidade Estadual de Campinas (UNICAMP), 2005.

[5] M. P. Trevizan, **Processamento Paralelo na Simulação de Campos Eletromagnéticos pelo Método das Diferenças Finitas no Domínio do Tempo**, Dissertação de Mestrado, Departamento de Engenharia de Telecomunicações e Controle, Escola Politécnica da Universidade de São Paulo, São Paulo, 2007

[6] R. P. Picanço, "**Desenvolvimento de uma Interface Integrada para o Projeto e Análise de Antenas Utilizando o Método das Diferenças Finitas no domínio do Tempo (FDTD)**", Dissertação de Mestrado, Departamento de Engenharia Elétrica, Universidade de Brasília / Faculdade de Tecnologia, 2006.

[7] Foster, I.; Kesselman, C. – **The Grid: Blueprint for a New Computing Infrastructure** 1st edition – Morgan Kaufman Publishers – San Francisco, USA – ISBN: 1555604758, Novembro, 1998

[8] Treadwell, J., – **Open Grid Services Architecture Glossary of Terms, version 1.6** – Open Grid Forum, GFD I.120 – edited by Treadwell, J., December, 2007

[9] Gabhart, K.; Bhattacharya, B. – **Service Oriented Architecture for Executives** – John Wiley & Sons – New Jersey, USA – ISBN: 978047026091-3, 2008

[10] Plaszczack, P.; Whellner, R. – **Grid Computing – The Savvy Manager's Guide** – John Morgan Kaufmann Publishers, 2006

[11] Li, J.; Ziegler, W.; Wäldrich, O.; Mallman, D. – **Towards SLA Based Software License Management in Grid Computing** – CoreGRID Technical Report– Institute on Resource Management and Scheduling, Junho, 2008

[12] Gorlatch, S.; Fragopoulou, P., Priol, T. – **Grid Computing – Achievements and Prospects** – Springer, 2008

[13] Senna, C. R., **GPO – Um middleware para orquestração de serviços em grades computacionais**, *Dissertação de Mestrado*, IC/UNICAMP, Campinas, Fevereiro de 2007.

[14] Grandinetti, L., – **Grid Computing: The New Frontier of High Performance Computing** – Advances in Parallel Computing, vol 14 – Elsevier Limited, ISBN: 9780444519993, 1999

[15] Romberg, M., – **The UNICORE Architecture Seamless Access to Distributed Resources – High Performance Distributed Computing**, 1999. Proceedings. The Eighth International Symposium on Volume , Issue , 1999 Page(s):287 - 293

[16] Sotomayor, B., Childers, L., **Globus Toolkit 4 – Programming Java Services**, Morgan Kaufmann - Elsevier, 2006

[17] **UniGrids** - http://www.unigrids.org/ - Acesso em 23/03/2009

[18] Yu, J., Buyya, R., – **A Taxonomy of Scientific Workflow Systems for Grid Computing** – SIGMOD Record, Vol. 34, No. 3 – September, 2005 IEEE International Symposium on Volume 2, Issue , 9-12 May 2005 Page(s): 686 - 693 Vol. 2

[19] **DEISA User Documentation for UNICORE** – Disponível em: http://www.deisa.eu/usersupport/user-documentation/unicore - Acesso em 23/03/2009

[20] Rambadt, M., Vanni, A., Niederberger, R., **Integration of GridFTP as an Alternative File Transfer in UNICORE for the DEISA Infrastructure**, - IEEE Seventh International Conference on eScience, Volume , Issue , 7-12 Dec. 2008 Page(s):516 – 523, 2008

[21] Wang, Y., Luan, Z., Qian, D., Huang, Y., Chen, T., Han, B., Ren, Y., Yu, K., Jiang, H., **DDGrid – A Grid Computing Environment with Massive Concurrency and Fault-tolerance Support**, IEEE Seventh International Conference on Grid and Cooperative Computing, 2008

[22] B. Schuller, B. Demuth, H. Mix, K. Rasch, M. Romberg, U. Maran, S. Sild, P. Bała, E. del Grosso, M. Casalegno, N. Piclin, M. Pintore, W. Sudholt and K. K. Baldridge, **"Chemomentum - UNICORE 6 based infrastructure for complex applications in science and technology"**, In: L. Bouge, M. Forsell, J. L. Traff, A. Streit,W. Ziegl (Eds.) *Euro-Par 2007 Workshops: Parallel Processing (Lecture Notes in Computer Science 4854)* Springer-Verlag Berlin Heidelberg 2008, pp. 94–103

[23] **OMII-Europe -** http://omii-europe.com/index.html – Acesso em 25/04/2009

[24] Menday, B., Hagemeier, Z., Fiameni, G., Cacciari, C., Melato, M., Curtoni, A., van den Berghe, S. - **An Easy Way to Access Grid Resources** - IEEE Computer Society - Proceedings of the 2008 12th Enterprise Distributed Object Computing Conference Workshops, 2008

[25] **Applications and Middleware-Support in VIOLA** http://www.viola-testbed.de/index.php?id=application – Acesso em 25/04/2009

[26] Schuller, B., – **UNICORE 6 - Architecture**– UNICORE Migration Workshop, FZ Jülich, Langen, 2008

[27] Riedel, M., Schuller, B., Mallmann, D., Menday, R., Streit, A., Tweddell, B., Memon, S, M., Memon, S, A., Demuth, B., Lippert, T., **Web Services Interfaces and Open Standards Integration into the European UNICORE 6 Grid Middleware** - IEEE Computer Society - Proceedings of the 2008 11th Enterprise Distributed Object Computing Conference Workshops, 2008

[28] OGSA-BES - http://www.unicore.eu/community/development/OGSA-BES/index.php - Acesso em: 23/03/2009

[29] **HPC-P -** https://forge.gridforum.org/sf/projects/ogsa-hpcp-wg – Acesso em 14/06/2009

[30] **Torque Resource Manager -** http://www.clusterresources.com/products/torque-resource-manager.php – Acesso em 25/04/2009

[31] **Tivoli LoadLeveler -**
http://www-03.ibm.com/systems/clusters/software/loadleveler/index.html – Acesso em 25/04/2009

[32] **Platform LSF -** http://www.platform.com/Products/platform-lsf-family – Acesso em 25/04/2009

[33] **SLURM -** https://computing.llnl.gov/linux/slurm/slurm.html – Acesso em 25/04/2009

[34] **OpenCCS -** https://www.openccs.eu/core/ – Acesso em 25/04/2009

[35] **Distributed Resource Management Application API -** http://drmaa.org/ – Acesso em 25/04/2009

[36] Riedel, M., Menday, R., Streit, A., Bala, P., **A DRMAA-based Target System Interface Framework for UNICORE –** IEEE Proceedings of the 12th International Conference on Parallel and Distributed Systems**,** 2006

[37] Menday, R., **UNICORE Gateway**, Manual de *Software* – 2008.

[38] **SAML -** http://saml.xml.org/ – Acesso em 14/06/2009

[39] Maguire, T., Snelling, D., Banks, T., – **Web Services Service Group 1.2** – OASIS Standard, Abril, 2006

[40] Foster, I., Lane, P., Lee, W., Newhouse, S., Pickles, S, U., Pulsipher, D., Smith, C., Theimer, M., – **OGSA – Basic Execution Service** – GFD-R.108, Open Grid Forum, Novembro, 2008

[41] Mach, R., Lepro-Metz, R., Jackson, S., McGinnis, L., **Usage Record – Format Recommendation** – GFD-R-P.098, Open Grid Forum, Setembro, 2006

[41] Godoy, R, M, B., "**Relatório de Arquitetura do *Software* SSAR-BR – Algoritmo de Geração de Malha Tridimensional para o Método Numérico FDTD**", DMO/FEEC/UNICAMP e CPqD, Dezembro, 2007.

[42] Eric Braude**, "Projeto de *Software* – Da programação à arquitetura: uma abordagem baseada em Java",** Bookman, 2005.

[43] Gamma, E., Helm, R., Johnson, R., Vlissides, J., "**Design Patterns, Elements of Reusable Object-Oriented Software**", Addison-Wesley, 1995.

[44] Deitel, P, J., **Java – How to Program**, 4th Edition, Prentice Hall, 2006

[45] Silva-Santos, C, H., Gonçalves, M, S., Ambrosio, L, A., Godoy, R, M, B., Freitas, I, J., Hernández-Figueroa, H, E., "**Relatório de Especificação da Base de Modelos do *Software* SSAR-BR**", DMO/FEEC/UNICAMP e CPqD, Outubro, 2007.

[46] Intel Software, "**Intel's Grid Programming Environment**", Whipe Paper, 2006.

[47] **Logmein -** https://secure.logmein.com/home.asp?hp=4 – Acesso em 26/08/2009

[48] M. Siegbahn, C. Törnevik, *Measurements and FDTD Computations of the IEEE SCC 34 Spherical Bowl and Dipole Antenna*, Ericsson, 2002.

[49] IEEE Standard, *IEEE Recommended Practice for Determining the Peak Spatial-Average Specific Absorption Rate (SAR) in the Human Head from Wireless Communications Devices: Measurement Techniques*, 2003.

[50] **Meep -** http://ab-initio.mit.edu/wiki/index.php/Meep - Acesso em 03/10/2010

[51] **ImageMagik -** http://www.imagemagick.org/script/index.php - Acesso em 03/10/2010

[52] **Octave -** http://www.gnu.org/software/octave/ - Acesso em 03/10/2010

[53] **VTK -** http://www.vtk.org/ -Acesso em 03/10/2010