Vinicius da Cunha Martins Borges

Uma Abordagem de Submissão e Monitoração de Múltiplas Tarefas para Ambientes de Grade Computacional Utilizando Dispositivos Móveis

Florianópolis - SC

2006

UNIVERSIDADE FEDERAL DE SANTA CATARINA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Vinicius da Cunha Martins Borges

Uma Abordagem de Submissão e Monitoração de Múltiplas Tarefas para Ambientes de Grade Computacional Utilizando Dispositivos Móveis

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação

Orientador:

Prof. Dr. Mario Antonio Ribeiro Dantas

Florianópolis, dezembro de 2006

Uma Abordagem de Submissão e Monitoração de Múltiplas Tarefas para Ambientes de Grade Computacional Utilizando Dispositivos Móveis

Vinicius da Cunha Martins Borges

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação Área de Concentração Sistemas de Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

Prof. Dr. Raul Sidnei Wazlawick

Coordenador do Curso

Banca Examinadora

Prof. Dr. Mario Antonio Ribeiro Dantas (Orientador)

Prof. Dr. Nelson Francisco Favilla Ebecken

Prof. Dr. Frank Augusto Siqueira

Prof. Dr. Roberto Willrich

Agradecimentos

A Deus, por iluminar-me e abençoar-me sempre.

Aos meus pais, Adebrair Martins Borges e Gelsimar da Cunha Martins Borges, pelo amor, carinho, presença em todos os momentos da minha vida e apoio incondicional em todos os meus sonhos.

Ao meu orientador, Mario Antonio Ribeiro Dantas, que tem minha eterna gratidão e admiração, pela excelente orientação, por sua dedicação e disponibilidade incansáveis, por suas idéias brilhantes, pela credibilidade depositada em mim, pela serenidade de nossas conversas que não me deixavam desanimar, por seus ensinamentos e pela parceria imprescindível para a realização desta dissertação.

Aos colegas do LaPeSD (Parra, Guilherme, Christopher, Jeferson, Denise, Alex e Rafael) pelos momentos de descontração, companheirismos, conselhos, amizade e sugestões que me impulsionaram para a realização desta dissertação.

Aos colegas Diogo, Henrique, Parra, Jeferson e Denise pelo tempo e esforço dedicado aos testes de funcionalidade que teve como objetivo melhorar a abordagem proposta nesta dissertação.

Ao amigo Tércio Filho (vulgo Tercim), amigo que pacientemente concedeu o aparelho celular utilizado em alguns experimentos realizados nesta dissertação e também pelas várias conversas e sugestões que me animaram durante o desenvolvimento deste trabalho, por quem tenho uma grande amizade, admiração e consideração.

Ao amigo Caetano, amigo que tem meu reconhecimento e gratidão, pela amizade e otimismo que me incentivaram e fizeram acreditar no valor do meu trabalho, pelas numeras suges-

tões, contribuições e conversas que me ajudaram a ter idéias, a criar e a melhorar a dissertação.

À Vera Sodré (Verinha) pela simpatia, prestatividade e paciência oferecida em tantos momentos de dúvidas.

À Melissa Lemos, pessoa que atenciosamente me auxiliou a desenvolver exemplos práticos de workflows usados nesta dissertação. Prova disto são os vários e-mails que lhe enviei e toda às vezes me respondeu pacientemente.

À Petrobrás pelo apoio financeiro parcial.

A todos que direta ou indiretamente contribuíram para a realização deste trabalho.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

AAB : Automatic Application Builder

AJO : Abstract Job Object

API : Application Programing Interface

c-HTML : Compact HTML

DAG : Directed Acyclic Graph

DAGMan : Directed Acyclic Graph Manager

EIS : Enterprise Information System

EPSRC : Engineering and Physical Sciences Research Council

GADL : Grid Application Definition Language

GPL : General Public License

GPRS : General Packet Radio Service

GRAM : Grid Resource Allocation Manager

GSFL : Grid Services Flow Language

GSI : Grid Security Infrastructure

GSH : Grid Service Handle

GSR : Grid Service Reference

GSS : Generic Security Service

GUI : Graphic User Interface

GTPL : Globus Toolkit Public License

GT : Globus Toolkit

GWES : Grid Workflow Execution Service

GWFE : Gridbus WorkFlow Engine

HTML : HyperText Markup Language

HTTP : HyperText Transfer Protocol

JVM : Java Virtual Machine

LGPL : Lesser General Public License

MDS : Monitoring and Discovery Service

MMJFS : Master Managed Job Factory Service

OGSA : Open Grid Services Architecture

OGSI : Open Grid Services Infrastructure

OV : Organização Virtual

OWL : Ontology Web Language

P2P : Peer-to-Peer

PDA : Personal Digital Assistants

QoS : Quality of Service

RLS : Replica Location Service

RPC : Remote Procedure Call

SCUFL : Simple Conceptual Unified Flow Language

SDK : Software Development Kit

SGW : Sistema de Gerenciamento Workflow

SOAP : XML Protocol / Simple Object Access Protocol

SOA : Service-Oriented Architecture

SRB : Storage Resource Broker

SSL : Secure Socket Layer

STR : Sistema de Tempo Real

TC : Transformation Catalog

TCP/IP : Transmission Control Protocol / Internet Protocol

TI : Tecnologia de Informação

UDDI : Universal Description Discovery Integration

UML : Unified Modeling Language

UNICORE : UNiform Interface to COmputing REsources

WAP : Wireless Application Protocol

WCT : Workflow Composition Tool

WML : Website Meta Language

WSDL : Web Services Description Language

WSFL : Web Services Flow Language

WSRF : Web Services Resource Framework

XML : Extensible Markup Language

Resumo

Ambientes de grade computacional são configurações reconhecidas para fornecer alto desempenho para vários tipos de aplicações, através do compartilhamento de uma grande escala de recursos geograficamente distribuídos. Por outro lado, dispositivos móveis são equipamentos interessantes para fornecer aos usuários acesso em qualquer hora e lugar a recursos, informações e serviços. Desta forma, a interação entre estes dois ambientes (isto é, computação em grade e computação móvel) está sendo considerada como uma solução eficiente por diversos usuários, para a obtenção de alto desempenho e fornecimento de uma maior mobilidade na execução de suas aplicações complexas.

Entretanto, devido ao crescente volume e distribuição dos dados, e também a necessidade de execução de inúmeras tarefas para tratar estes dados, tornam-se cada vez mais complexos os processos e aplicações para a resolução de problemas a partir de dispositivos móveis. A execução coordenada e combinada de várias tarefas, que trabalham juntas acessando esta grande quantidade de dados, pode facilitar na solução destes problemas complexos que utilizam a infraestrutura contida em configurações de grade. No entanto, na maioria das pesquisas relacionadas existe a possibilidade de submissão e monitoração de uma tarefa por vez na interface do dispositivo. Nestas, usuários precisam controlar e ordenar a submissão destas tarefas que trabalham juntas para resolver um problema em uma configuração de grade. Desta forma, a falta de um mecanismo automatizado que também forneça uma submissão e monitoração coordenada e organizada dessas várias tarefas torna mais difícil para os usuários de dispositivos móveis resolverem seus problemas, utilizando o poder computacional disponibilizado pela grade.

Com o objetivo de proporcionar uma forma melhorada de submissão e monitoração de várias tarefas em grades computacionais, a partir do dispositivo móvel, esta dissertação apresenta como alternativa a utilização do mecanismo de *workflow*. Este mecanismo oferece uma abordagem diferencial que permite aos usuários de dispositivos móveis submeterem e monitorarem suas aplicações em ambientes de grade de uma maneira automatizada e coordenada. Um protótipo foi projetado e implementado para validar a proposta.

Em nossos experimentos, o mecanismo de *workflow* provou ser eficiente. A utilização do *workflow* nestes dispositivos, nos permitiu concluir que a junção destas duas áreas traz melhorias para os usuários destes aparelhos, permitindo uma melhor utilização da configuração de grade. Exemplos das melhorias alcançadas com a adoção desta abordagem são: uma maior agilidade e um menor consumo de energia de bateria para submeter e monitorar várias tarefas do que a forma utilizada nas pesquisas relacionadas. Exemplos de *workflows* de bioinformática foram empregados, onde usuários foram capazes de acompanhar passo a passo seu progresso de execução transparentemente. Por fim, foram desenvolvidas novas funcionalidades e informações detalhadas da execução *workflow*.

Palavras-chave: *workflow*, grades computacionais, dispositivo móvel, submissão e monitoração de tarefas

Abstract

Grid environment are well known as an interesting configuration that provides high performance to several applications types through the sharing of a large number of resources distributed geographically. On the other hand, mobile devices are interesting equipments to provide to the users access anywhere and anytime to the resources, information and services. Therefore, the interaction among these two environments (i.e, grid computing and mobile computing) it is being currently considered as an efficient solution by several users, for obtaining high performance and providing more mobility in the execution of their complex applications.

However, the crescent volume and the distribution of data, and also the countless tasks to treat these data, turn more complex the retrieval of information that aid in the resolution of problems from the mobile devices. The coordinated and combined execution of several tasks can facilitate in the solution of these complex problems that use to the infrastructure of the grid configurations, where these tasks work together to access this great quantity of data. However, most of the related researches allows to submit and to monitor a task per time in the interface of the device. In these, users need to control and to order the submission of these tasks that work together to solve a problem in a grid configuration. Therefore, the lack of an automated mechanism that also provides a coordination and organization in the submission and monitoring of those several tasks turns more complex of problems resolution for the users of mobile devices, utilizing the computational power of the grid.

With the objective of providing an improved way of submission and monitoring of several tasks in grid computing from the mobile device, this dissertation presents as alternative the use of the workflow mechanism. This mechanism offers an differential approach that allows users of mobile devices to submit and monitor their applications in a grid environment in an automated and coordinate way. A prototype was projected and implemented to validate our objective.

In our experiments, the workflow mechanism proved to be efficient. The research done with relationship the representation form of workflow in these devices, allowed to conclude that the junction of these two areas brings great benefits to the users, because it allows better utilization of the grid configuration. Examples of the improvements reached with the adoption of this approach are: better agility for problem resolution and it also enables to consume less battery energy to submit and monitor several tasks than the manner used in the most related research. Examples of bioinformatic workflows were used, where users were capable to accompany step to step the progress of execution transparently. Finally, new functionalities and detailed information of the workflow execution were provided.

Keywords: workflow, grid computing, mobile device, tasks submission and monitoring

1 Introdução

Grades computacionais são ambientes geograficamente distribuídos que permitem o compartilhamento de recursos diversificados e em larga escala. Estes recursos são geralmente fornecidos por organizações virtuais (indivíduos ou instituições) que determinam quem e de que forma esses recursos podem ser utilizados dentro da configuração grade. Estas organizações são baseadas em políticas de utilização definidas em um acordo entre o fornecedor e consumidor de recursos (FOSTER; KESSELMAN; TUECKE, 2001). Desta forma, o ambiente de grade computacional pode fornecer alto desempenho para resolução de problemas complexos, onde estes problemas necessitam de alto poder computacional, grande quantidade de recursos e também a execução coordenada de várias tarefas que trabalham juntas para resolvê-los (FOSTER; KESSELMAN, 2003).

Por outro lado, os dispositivos empregados na computação móvel enfrentam um conjunto de restrições de recursos (por exemplo, baixo poder de processamento e armazenamento). Estas restrições impõem uma grande dificuldade para que dispositivos móveis possam fornecer aplicações, ou serviços, que envolvem resolução de problemas complexos requisitando alto poder computacional. No entanto, a computação móvel tem como função principal oferecer ao usuário acesso a informações, aplicações, recursos e serviços em qualquer hora e lugar. Através de dispositivos móveis tais como PDAs e telefones celulares, a infra-estrutura de redes sem-fio podem fornecer tais facilidades (FORMAN; ZAHORJAN, 1994).

Nesta ótica, a interação dos dispositivos móveis com o ambiente de grade oferece vantagens aos usuários que utilizam estes dispositivos. Exemplos são: usufruir a grande quantidade de recursos disponibilizados nas configurações de grade, com o intuito de obter melhor desem-

penho na execução de processos complexas; determinar parâmetros de QoS para a execução de aplicações; acessar serviços de grade em qualquer lugar e hora, assim tornando mais rápido e fácil o acesso às informações produzidas pela configuração de grade de forma remota (PHAM; HUANG; DULAN, 2002), (LITKE; SKOUTAS; VARVARIGOU, 2004) e (KURKOVSKY; BHAGYAVATI; YANG, 2004).

Várias pesquisas têm permitido a interação entre dispositivos móveis e *middleware* grade, exemplos são mostrados em (AHMAD et al., 2004; MCKNIGHT; HOWISON; BRADNER, 2004; GONZALEZ-CASTANO; VALES-ALONSO; LIVNY, 2003; CLARKE; HUMPHREY, 2002; BROOKE; PARKIN, 2005). Entretanto, a maioria destas pesquisas permite submeter e monitorar uma tarefa por vez, onde usuários móveis precisam controlar e ordenar a submissão destas tarefas que trabalham juntas para resolver um problema em uma configuração grade. Todavia, usuários podem submeter tarefas em uma ordem errada para resolver o problema e também podem ocorrer atrasos na submissão destas tarefas por alguma razão. Portanto, um mecanismo mais automatizado para submissão e monitoração de várias tarefas que cooperam entre si para resolver um único problema fornece uma interface melhorada (isto é, mais ágil, organizada e controlada) para usuários de dispositivos móveis.

Para este propósito, foi utilizado nesta dissertação o mecanismo *workflow* (HOLLINGSWORTH, 1995) como uma alternativa mais automatizada para execução de várias tarefas a partir de um dispositivo móvel no ambiente de grade. *Workflows* são empregados em configurações de grade recentemente, como mostrado em (CANNATARO et al., 2004; OINN et al., 2004; SHAHAB et al., 2004), e podem fornecer vantagens para usuários móveis, como por exemplo: devido a automatização deste mecanismo ele permite um aumento de agilidade na execução de várias tarefas que trabalham juntas para resolver um problema, minimização de erros na ordenação das submissões de várias tarefas (devida, à organização realizada pelo *workflow*), submissão de aplicações que são organizadas em recursos distribuídos; e execução distribuída em múltiplas organizações virtuais para obter capacidades específicas de processamento.

Para muitas pessoas e instituições, a atração de computação móvel é baseada no fato que

pode ser possível alcançar um ganho de produtividade fora dos locais de trabalho habituais, como por exemplo: um biólogo pode desejar executar um sequenciamento de DNA de certa espécie, um físico pode querer analisar os dados gerados a partir do ambiente de grade e um químico pode também calcular os riscos ambientais no transporte de poluentes na atmosfera iniciados por um acidente. Desta forma, alguns *workflows* (por exemplo, bioinformática) que levam horas ou até mesmo dias para concluírem sua execução (MYGRID, 2006), mesmo com o ambiente de alto desempenho fornecido pela grade, podem ser submetidos e monitorados a partir de locais remotos através de dispositivos móveis e também ter os resultados finais ou parciais. Além disso, várias tarefas que eram coordenadas manualmente pode ser automatizadas com o advento da tecnologia *workflow*.

A partir das vantagens da interação dos dispositivos móveis com a grade computacional e o uso de *workflows*, esta dissertação tem como objetivo principal apresentar uma abordagem de utilização de *workflows* para automatizar, organizar e combinar a execução e monitoração de várias tarefas a partir de dispositivos móveis, usufruindo o alto poder computacional oferecido pela grade computacional. Além disso, tornando certos detalhes transparentes aos usuários de dispositivos móveis (como por exemplo, gerenciamento de recursos, movimentação de dados, segurança, entre outros), fornecendo também interfaces com mais informações detalhadas e/ou mais funcionalidades do que os trabalhos anteriores fornecem e por último, implementar interfaces adequadas às restrições de cada tipo de dispositivo móvel.

Para realização desta dissertação, primeiramente foi realizada uma investigação para encontrar uma forma mais adequada aos dispositivos móveis para submissão e monitoração de *workflows*, com objetivo de fornecer uma interface adaptada as restrições destes aparelhos. O próximo passo, foi o desenvolvimento de um componente de gerenciamento necessário para execução de *workflows*. Este componente se encontra em um servidor tradicional, está preparado para comunicar com os dispositivos móveis, receber os pedidos de submissão de *workflows* e ter a capacidade de acessar os serviços e recursos disponibilizados no *middleware* de grade de forma mais transparente possível ao usuário móvel. Na continuação da pesquisa apresentada na dissertação, foi desenvolvida uma interface que fornece várias funcionalidades (por exemplo,

submissão e monitoração), a qual foi representada como um portal contido nos dispositivos móveis, visando tornar a aceitação da proposta apresentada mais simples e clara. Além disso, foi realizada uma análise comparativa do tempo total de execução na resolução de um problema e também o consumo de bateria dos dispositivos, ambos os aspectos são comparados com a abordagem usada na maioria das pesquisas estudadas. Ao final da pesquisa, foi empregada técnicas de engenharia e avaliação de usabilidade com intuito de melhorar as interfaces desenvolvidas.

Nesta dissertação, os capítulos 2, 3 e 4 apresentam uma revisão da literatura necessária para a realização do trabalho. O capítulo 2 trata do tema Grades Computacionais, fornecendo a base teórica para compreensão destes ambientes e do presente trabalho. O capítulo 3 apresenta aspectos importantes dentro do contexto de *workflows*, traçando a base teórica necessária para entendimento e desenvolvimento de aplicações baseadas em *workflow*. No capítulo 4 são apresentados os principais trabalhos relacionados de interação da grade computacional com os dispositivos móveis.

O capítulo 5 descreve detalhes do trabalho realizado nesta dissertação, apresentando como foram desenvolvidos os componentes da abordagem proposta; justificando a escolha das ferramentas utilizadas para desenvolvimento em dispositivos móveis e interação com o *middleware* grade; mostrando a maneira como foi desenvolvida a interface do Portal respeitando as restrições de cada tipo de dispositivo móvel utilizado neste trabalho e por último apresentando uma comparação entre a abordagem aqui proposta e as demais abordagens anteriores, analisando principalmente os aspectos de desempenho e utilização de energia da bateria destes dispositivos móveis.

Finalmente, no capítulo 6 são apresentadas as conclusões e resultados obtidos durante a realização desta dissertação, juntamente com algumas sugestões para trabalhos futuros.

2 *Grades Computacionais*

Segundo (DANTAS, 2005), ambiente de grade computacional é a forma mais recente do processamento geograficamente distribuído que tem como característica a grande infra-estrutura das redes, empregada para compartilhamento de vários recursos heterogêneos (servidores, aglomerados de computadores, programas, dados e serviços computacionais) pertencentes a diferentes organizações que compõem a configuração do ambiente de grade computacional. Estas organizações são conhecidas como organizações virtuais e são entidades que compartilham recursos sob uma determinada política ou regra em uma configuração de grade (por exemplo, empresas, centro de pesquisas e universidades) (FOSTER; KESSELMAN; TUECKE, 2001).

Grade computacional tem como objetivo o fornecimento de grande poder computacional para resolver problemas complexos de uma grande classe de aplicações científicas, industriais e comerciais. Sendo assim, alcançando a interoperabilidade entre as organizações virtuais, através da habilidade de coordenação, cooperação, agregação e compartilhamento de recursos computacionais distribuídos geograficamente (DANTAS, 2005), (FOSTER; KESSELMAN, 2003).

O termo grade, em inglês *grid*, tem sua origem segundo (FOSTER; KESSELMAN, 1999), no termo *electrical power grid*, que significa rede elétrica, onde o usuário utiliza energia sem se preocupar com a fonte geradora e distribuída da mesma. De maneira semelhante, grade computacional permite o compartilhamento de recursos em larga escala de forma transparente ao usuário da grade através da conexão destes recursos por meio de uma rede de computadores (por exemplo, Internet). Assim, todos os recursos existentes em cada ponto de conexão são somados e vistos como se estivessem totalmente unidos e passando para o usuário a imagem de um único ambiente de computação de alto desempenho.

De forma geral o conceito de grade é apresentado em (BUYYA, 2002):

> *"Grade é um tipo de sistema paralelo e distribuído o qual permite compartilhamento, seleção e agregação dinâmica em tempo de execução de recursos autônomos geograficamente distribuídos, dependendo de sua disponibilidade, capacidade, desempenho, custo e requisitos de qualidade de serviço necessários aos usuários."*

Segundo (DANTAS, 2005), grades computacionais podem ser consideradas também sob o paradigma de melhoria de alguns aspectos físicos nas redes de comunicação e computadores, tais como:

- Melhor utilização de largura de banda;
- Utilização agregada de um grande poder computacional;
- Acesso rápido a dados, pacotes de software e dispositivos remotos com qualidade de serviço (QoS);
- Melhor utilização de processadores remotos, memórias e espaço em discos.

Problemas complexos que possivelmente levariam uma quantidade de tempo muito elevada, devido à limitação de processamento dos recursos locais, podem agora ser resolvidos com mais rapidez através da utilização de grade. Estes problemas geralmente requisitam uma quantidade enorme e diversificada de recursos (por exemplo, alto poder de processamento, grande espaço para armazenamento de dados e/ou a utilização de dispositivos científicos sofisticados) que estão distribuídos em diferentes lugares e com diferentes políticas de acesso. Sendo assim, a utilização das grades computacionais possibilita a agregação de grande quantidade de recursos que possibilita resolver estes problemas de forma mais rápida, uma vez que a grade computacional permite o uso de vários recursos em locais remotos através de uma rede de computadores (FOSTER et al., 1997).

(SKILLICORN, 2002) apresenta as seguintes motivações para o uso de ambientes de grade computacionais:

- Necessidade de paralelismo: O uso de ambientes de grade devido à necessidade de paralelismo é justificado devido à grande paralelização proporcionada por este;
- Necessidade de ir além de um sistema paralelo isolado: Sistemas paralelos isolados, muitas vezes não proporcionam o desempenho desejado, logo o uso de ambientes de grade pode proporcionar este desempenho;
- Tem um custo menor que sistemas monolíticos.

De forma resumida, as principais características de grade computacional são mostradas em (FOSTER; KESSELMAN; TUECKE, 2001), tais como:

- Possuem grande porte, devido à grande quantidade de recursos disponíveis neste ambiente;
- São compostas por várias organizações virtuais, onde cada organização virtual fornece recursos que são compartilhados no ambiente de grade de forma controlada. Desta forma, a grade atravessa as fronteiras de uma organização humana, determinando as políticas de acesso e de uso dos recursos compartilhados e permitindo que os membros possam colaborar um com o outro para alcançar um objetivo comum (isto é, a resolução de um único problema);
- São geograficamente distribuídos, portanto a latência envolvida na movimentação dos dados entre os seus recursos é considerável, sendo este fator predominante nas aplicações;
- São dinâmicos, pois os seus recursos disponíveis sofrem mudanças na mesma escala de tempo que a duração de uma aplicação;
- São heterogêneos, ou seja, a estrutura de hardware e de software disponibilizada pelas organizações virtuais que compõe a grade difere de diversas maneiras;

- Fornecem interoperabilidade entre diferentes organizações, uma vez que nestes ambientes é necessário assegurar que os relacionamentos de compartilhamento podem ser iniciados entre partes arbitrárias, acomodando novos participantes dinamicamente, através de diferentes plataformas, linguagens e ambientes de programação. Similarmente aos ambientes de redes, interoperabilidade em grade é alcançada com a utilização de protocolos comuns.

2.1 Áreas de Aplicação

Várias áreas e aplicações exigem compartilhamento, de recursos em larga escala para obter características como: alto desempenho, alta disponibilidade, alta taxa de transmissão de dados, armazenamento e processamento para grande quantidade de dados, emprego de sofisticados dispositivos científicos e outras vantagens. Entretanto existem algumas áreas de destaque, nas quais a utilização de grades se torna realmente significativa. (DANTAS, 2005) e (FOSTER; KESSELMAN, 1999) através de pesquisas experimentais, identificaram as seguintes áreas onde é interessante a utilização de grades computacionais, são:

- **Supercomputação Distribuída**: as aplicações que utilizam esta abordagem são as aplicações que não seria viável executar somente com a capacidade de processamento de um único sistema computacional. Sendo assim, estas aplicações exigem requisitos computacionais tão altos que somente é possível atendê-los através da agregação de múltiplos recursos de alta capacidade, os quais são encontrados em um ambiente de grade, formando um único supercomputador virtual (FOSTER; KESSELMAN, 1999). Alguns exemplos desta abordagem são: o sistema NetSolve (CASANOVA; DONGARRA, 1995), o qual consiste de uma aplicação cliente-servidor projetada para tornar simples a resolução de problemas, relacionados a ciência da computação, através da rede, e oferece vários tipos de interfaces, permitindo que usuários utilizando diferentes tipos de programas (C, Fortran, MATLAB ou WWW) utilizem o NetSolve; DIS (*Distributed Interactive Simulation*) que é uma técnica de simulação empregada em planejamento militar, ensino e também aplicações de previsão do tempo e cosmologia;

- **Computação de Alto-Desempenho**: esta área é voltada às aplicações que requerem rápida velocidade no processamento e transferência de grande quantidade de dados. Nestas aplicações, segundo (FOSTER; KESSELMAN, 1999), ambientes de grade são usados para escalonar um grande número de tarefas fracamente acopladas, com o objetivo de colocar ativos computadores que não possuem muita utilização, isto é, freqüentemente ociosos. Exemplos clássicos desta abordagem são: Condor High-Throughput (THAIN; TANNENBAUM; LIVNY, 2005) e o LSF (*Load Share Facility*) (ZHOU et al., 1993). Condor foi desenvolvido na Universidade de Wisconsin, e tem como objetivo maximizar a utilização de um conjunto de centenas computadores em diferentes laboratórios e espalhados pelo mundo com o mínimo de interferência entre as tarefas escalonadas e as atividades de quem os utiliza, identificando computadores ociosos e escalonando tarefas sobre eles;

- **Computação Sob-Demanda**: aplicações que necessitam fazer uso de recursos computacionais de forma intensiva e estes recursos não se encontram disponíveis localmente. Estes recursos podem ser classificados como repositórios de dados, pacotes de software, arquivos e poder computacional. A diferença entre as aplicações de supercomputação distribuída e esta abordagem é relacionada com o custo do desempenho. A computação sob-demanda utiliza os recursos de maneira colaborativa, apenas empregando recursos não locais que podem ser solicitados de forma distribuída para aumentar o desempenho. Já a supercomputação distribuída utiliza todos os recursos não locais para aumentar o desempenho. Como exemplo de utilização de computação sob-demanda pode-se citar o sistema desenvolvido na *Aerospace Corporation*, o qual possui como objetivo o processamento de dados para satélites meteorológicos (LEE; KESSELMAN; SCHWAB, 1996).

- **Computação para Grande Quantidade de Dados**: aplicações nesta abordagem utilizam processamento intensivo de grande quantidade dados, os quais estão geograficamente distribuídos. Exemplos bem difundidos desta abordagem são bibliotecas digitais e banco de dados distribuídos. Pode-se citar o projeto *Digital Sky Survey* que tem como objetivo a integração de dados digitais extraídos de analise espaciais de imagens digitais em grandes áreas do espaço (MOORE et al., 1999).

- **Computação Colaborativa**: as aplicações que fazem parte desta abordagem têm como objetivo proporcionar melhorias nas formas com que ocorrem interações humanas, sendo que as aplicações podem ser compartilhadas sem existência de altos custos (DANTAS, 2005) e (FOSTER; KESSELMAN, 1999). Um exemplo desta área de aplicação é o projeto NICE (*Narrative-based, Immersive, Constructionist/Collaborative Environments*), que consiste de um ambiente colaborativo para o aprendizado de crianças, permitindo que as crianças juntas aprendam conceitos sobre a natureza, estando elas ou não situadas no mesmo local (JOHNSON et al., 1998).

2.2 Arquiteturas

Para que o compartilhamento de recursos seja realizado da melhor forma possível entre diferentes organizações virtuais é necessário o estabelecimento de um acordo ou contrato entre consumidores e fornecedores de serviços, definindo de forma clara e cuidadosa quais recursos devem ser compartilhados, a quem é permitido o compartilhamento e as condições sobre as quais o compartilhamento pode ocorrer (FOSTER; KESSELMAN; TUECKE, 2001). Sendo assim, a arquitetura de um ambiente de grade pode ser visualizada como sendo **orientada a serviços**.

Nesta arquitetura identifica-se a interoperabilidade nas aplicações de OVs como questão central do ambiente, onde interoperabilidade significa a utilização de protocolos comuns para negociação de recursos, estabelecendo e gerenciando relações de compartilhamento (DANTAS, 2005). Além disso, os protocolos possuem fundamental participação para obtenção de interoperabilidade, pois a sua definição especifica como os elementos do sistema distribuído irão interagir entre si, determinando aspectos no ambiente de grade, como por exemplo: autenticação, autorização, mecanismos de troca de mensagens, compartilhamento de recursos, escalonamento e balanceamento de tarefas. Consequentemente, uma arquitetura de grade é em primeiro lugar uma arquitetura de protocolos (FOSTER; KESSELMAN; TUECKE, 2001).

Uma arquitetura de protocolos aberta e padronizada permite a extensibilidade, a interoperabilidade e portabilidade, facilitando a definição de serviços padrões que forneçam capacidades

aprimoradas. Desta forma, (FOSTER; KESSELMAN; TUECKE, 2001) apresenta uma arquitetura grade composta em camadas, de forma semelhante à arquitetura de protocolos TCP/IP, a arquitetura grade foi inicialmente organizada em camadas, constituída por uma pilha de protocolos. Na Figura 1 pode-se observar a arquitetura de um ambiente de grade e sua relação com a arquitetura de protocolos da Internet, ambas organizadas em camadas.

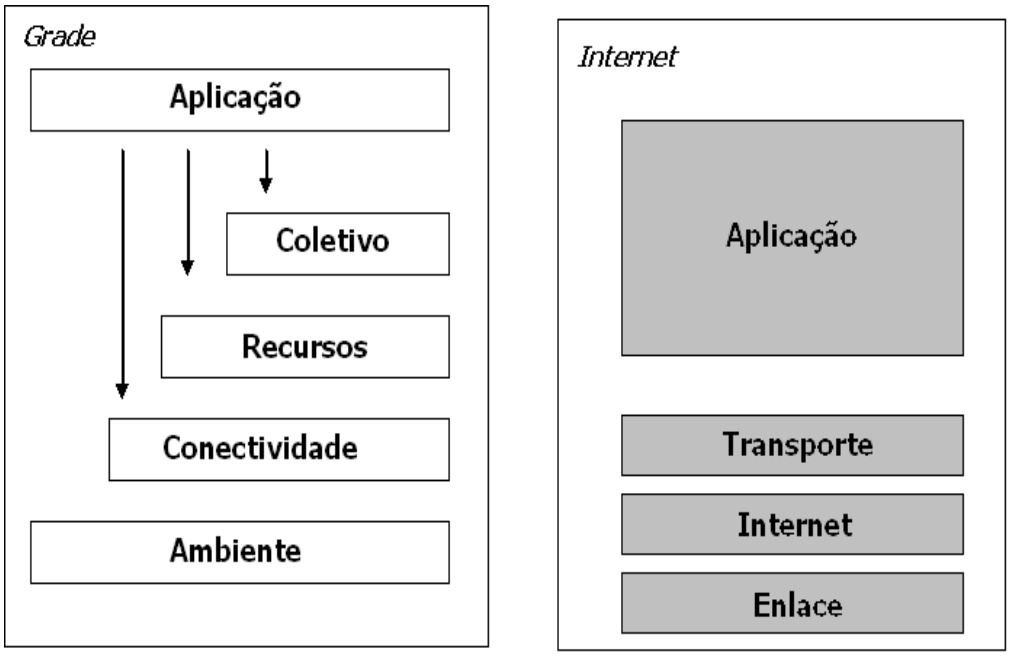

Figura 1: Relação entre Arquitetura Grade e Internet (DANTAS, 2005).

As camadas apresentadas na Figura 1 podem ser entendidas como (DANTAS, 2005):

- **Ambiente**: esta camada implementa operações específicas locais que ocorrem em cada recurso como resultado das operações de compartilhamento nos níveis superiores. De um lado, devem ser implementados mecanismos de negociação que façam uma solicitação para obtenção de informações sobre a estrutura, o estado e as possibilidades dos recursos. Do outro lado, mecanismos de gerenciamento de recursos devem fornecer formas de monitorar a qualidade de serviço (QoS);

- **Conectividade**: esta camada define os protocolos básicos de comunicação e autenticação de transações específicas da grade. Os protocolos de comunicação permitem a troca de dados entre os recursos existentes na camada ambiente, enquanto que os protocolos de au-

tenticação constroem os serviços de comunicação, provendo mecanismos criptográficos para verificação da identidade de usuários e recursos;

- **Recursos**: nesta camada são definidos os protocolos, APIs e SDKs que garantem segurança durante as negociações, iniciação, monitoração, controle, criação de relatórios, entre outras tarefas voltadas ao compartilhamento de recursos individuais. Implementações dos protocolos desta camada chamam funções pertencentes à camada ambiente para obterem acesso e controle aos recursos locais. Estes protocolos preocupam-se apenas com recursos individuais, ignorando questões a respeito de estado global e coleções distribuídas, sendo estas tratadas pela camada coletivo;

- **Coletivo**: enquanto no nível de recursos são tratadas as operações no âmbito de cada recurso individualmente, nesta camada os componentes atuam nas interações entre coleções de recursos. Os componentes dessa camada baseiam-se nas camadas recursos e aplicação e implementam uma grande variedade de serviços tais como:

  1. Serviços de diretório que permitem aos membros de uma organização virtual descubrirem a existência das propriedades de recursos;
  2. Serviços de autorização comunitários que reforçam a política de acesso aos recursos.

- **Aplicação**: esta camada compreende as aplicações dos usuários que estão ligados a uma organização virtual, isto é, os programas de aplicação para os quais usuários necessitam dos recursos, oferecidos pela grade, para que sejam corretamente executados. As camadas anteriores provêem serviços úteis às aplicações desenvolvidas que as invocam.

2.2.1 OGSA

Nos últimos anos, diversas ferramentas e aplicações foram implementadas na área de grade (por exemplo, Globus Toolkit (GLOBUS, 2006), UNICORE (ERWIN; SNELLING, 2001), Condor (THAIN; TANNENBAUM; LIVNY, 2005) e OGSI.NET (WASSON et al., 2004) por diferentes indivíduos e instituições levando fatalmente a falta de interoperabilidade entre as mesmas. Sendo

assim, a padronização tornou-se um elemento fundamental para o ambiente de grade, sendo necessário que pesquisadores da área de ciência da computação e desenvolvedores de aplicações unissem esforços para o desenvolvimento de uma arquitetura padronizada e aberta que oferecesse interoperabilidade entre serviços e recursos de forma mais natural (DANTAS, 2005).

Com o objetivo de padronização, pesquisadores do projeto Globus (GLOBUS, 2006) e da IBM (IBM, 1998) uniram esforços para propor o uso de um padrão com serviços orientados a Web conhecido como *Open Grid Services Architecture* (OGSA) (TALIA, 2002). Para alcançar um conjunto extensível de serviços que OVs possam agregar em vários modos, OGSA define o conceito de serviço de grade, em inglês *Grid service*, baseado em princípios e tecnologias de ambos grade computacional e *Web Services* (CURBERA et al., 2002).

De maneira semelhante a *Web Services*, a OGSA permite a interoperabilidade e portabilidade de hardware e software entre serviços fracamente acoplados. OGSA define um mecanismo padrão para a criação, atribuição de nome, descoberta persistente e transiente de instâncias de serviços de grade (TALIA, 2002). Para possibilitar isto, OGSA define um conjunto de extensões na linguagem padrão chamado WSDL, usando padrões abertos e consolidados como SOAP, XML e HTTP para criação e disponibilização de serviços de grade.

As operações dos serviços de grade são definidas em interfaces que correspondem ao WSDL *portType*. Cada *portType* está relacionado a um ou mais operações, na qual um cliente pode invocar através da troca de seqüências especificas de mensagens com a instância de serviço. Exemplos de operações são: *FindServiceData*, *SetTerminationTime*, *CreateService*, *RegisterService* e *FindByHandle*.Outras operações podem ser encontradas em (TALIA, 2002).

2.2.2 OGSI

A *Open Grid Services Infrastructure* (OGSI) (TUECKE et al., 2002) é a especificação técnica de extensões e especializações expressadas em WSDL, definindo padrões de interface, ações e esquemas para grades computacionais como requisitado pela abordagem OGSA. Estas interfaces ajudam a definir como construir, gerenciar e expandir um serviço de grade.

Diferentemente de *Web Services*, OGSI suporta a instância de serviços transientes, que podem ter uma duração curta e também pode ser criados e destruídos dinamicamente. Assim, um serviço de grade é um *Web Service* transiente baseado em protocolos de grade e descrito em WSDL.

Segundo (TUECKE et al., 2002), a especificação OGSI define dois níveis de esquema de nomes baseada nos serviços de apontadores e referência que são *Grid Service Handle* (GSH) e *Grid Service Reference* (GSR). Cada GSH é um identificador global para um serviço de grade determinado para toda a sua existência. Por outro lado, GSH não contém informações de como comunicar com o serviço de grade (por exemplo, quais métodos ele tem e quais tipos de mensagens ele aceita). Para solucionar este problema, o GSR foi criado como sendo um arquivo WSDL (isto é, dizendo quais métodos tem e quais os parâmetros de entrada e saída dos métodos) que fornece todas as informações necessárias para que os clientes possam comunicar com os serviços de forma correta.

2.2.3 WSRF

De acordo com (SOTOMAYOR, 2005), OGSI foi criado com a esperança que o mesmo eventualmente convergisse os serviços de grade em padrões *Web Services*. Entretanto, serviços de grade especificados pela OGSI apresentaram várias desvantagens que tem impedido esta convergência, por exemplo:

- A especificação OGSI é longa e densa;
- OGSI não trabalha bem com as ferramentas *Web Services* atuais;
- Muitos sistemas *Web Services* tem implementações orientadas a objetos, o próprio *Web Service* não está implementado para ser orientado a objeto. OGSI, por outro lado, pega muitos conceitos de orientação a objeto.

Para superar as desvantagens citadas acima e melhorar os serviços de grade no propósito de convertê-los em *Web Services*, um novo padrão conhecido como *Web Services Resource*

Framework (WSRF) (WSRF, 2004) foi apresentado para substituir o OGSI. A diferença entre a forma de especificação de serviços no Globus versão 3 e 4 é ilustrado na Figura 2, onde mostra os relacionamentos entre OGSA, OGSI e Grid Services no globus versão 3 e os relacionamentos entre OGSA, WSRF e *Web Service Stateful* no Globus versão 4.

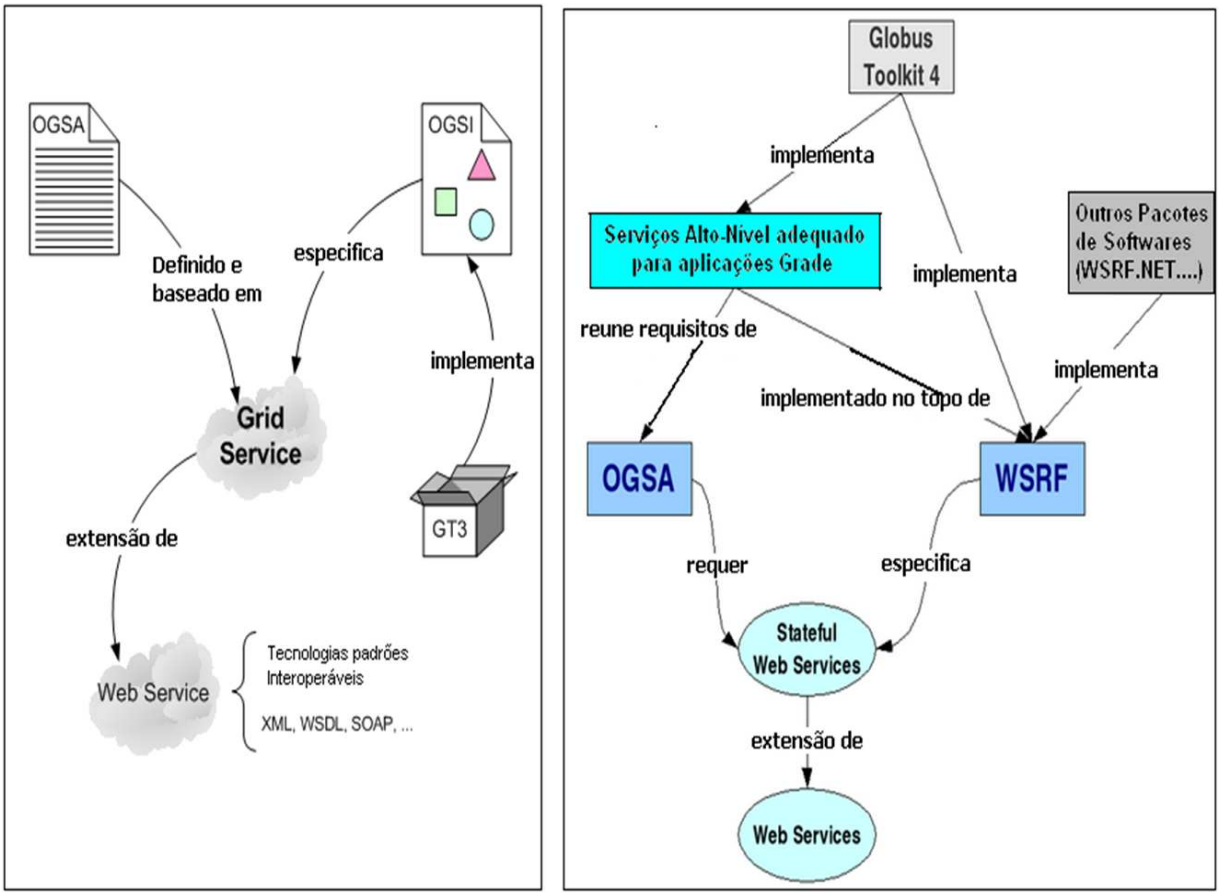

Figura 2: Especificação de Serviços em Globus Toolkit versões 3 (SOTOMAYOR, 2003) e 4 (SOTOMAYOR, 2005).

Com esta mudança, OGSA será baseado diretamente em *Web Services* ao invés de ser baseado em serviços de grade OGSI. A abordagem OGSA não é afetada pela mudança de padrão para WSRF. Todos os serviços de alto nível definidos em OGSA continuarão tendo as mesmas interfaces e especificações. A única diferença é que o *middleware* base será puramente *Web Services*. Esta mudança entrou em vigor na versão 4 do pacote de software para implementação de grades computacionais conhecido como Globus Toolkit (GLOBUS, 2006).

3 *Workflow*

Com o advento das grades computacionais, fornecendo vários serviços, recursos e a capacidade de resolver problemas cada vez mais complexos, está se tornando cada vez mais comum a execução cooperativa de múltiplas tarefas para a resolução de um único problema. Em geral, este conjunto de múltiplas tarefas possui interações, dependências entre as mesmas e requerem a utilização de várias ferramentas computacionais (aplicações e/ou grande banco de dados) que estão compartilhadas pelo ambiente de grade. Desta forma, estas tarefas representam um fluxo de trabalho onde dados são passados entre as várias tarefas obedecendo a certas regras. Sendo assim, torna-se necessária a utilização de uma forma para controlar e organizar este fluxo de execução de tarefas.

Nesta ótica, o conceito de *workflow* foi empregado neste trabalho de pesquisa, como uma alternativa para automatização e coordenação da execução destas múltiplas tarefas em ambientes de grade envolvidas na resolução de um único problema, apresentando-se com uma solução de granularidade mais fina e genérica. De forma geral o conceito de *workflow* é apresentado em (HOLLINGSWORTH, 1995):

> *"Workflow está relacionado à automatização de procedimentos, onde documentos, informação ou tarefas são passadas entre os participantes de acordo com um conjunto pré-definido de regras, para se alcançar ou contribuir para um objetivo global de um negócio. Apesar de um workflow poder ser manualmente organizado, na prática a maioria dos workflows são organizados dentro de um contexto de um sistema de informação para prover um apoio automatizado aos procedimentos"*

Uma definição de *workflow* em ambientes de grade é apresentado por (LASZEWSKI; HATEGAN, 2005). Semelhante a outros mecanismos formais, um *workflow* de grade computacional pode ser entendido como um conjunto de recursos e serviços de grade, uma expectativa de qualidade definida pelo(s) usuário(s) e um modelo *workflow* agindo neles. A seguinte função mostra a representação de um *workflow* de grade:

$$W_i = (G_r, G_s, Q_u, W_m) \quad . \tag{3.1}$$

Onde W_i = instanciação do *Workflow*, G_r = recursos Grade, Q_s = serviços Grade, Q_u = expectativa de qualidade do usuário e W_m = modelo *Workflow*.

Na literatura, Wi é também referenciada como um *workflow* concreto ou executável. Por outro lado, Wm é chamado de *workflow* abstrato. Segundo (YU; BUYYA, 2005), *workflow* concreto liga tarefas ou atividades de um *workflow* a recursos específicos, ao contrário do *workflow* abstrato, no qual o *workflow* é especificado sem referenciar recursos específicos para execução da tarefa.

A abordagem *workflow* tem mostrado uma tendência crescente em uso de muitos projetos de pesquisa em grades computacionais, como acontece em (JUNG et al., 2004), (YANG; LIANG; XU, 2005), (TAYLOR et al., 2005) e (YU; BUYYA, 2005). *Workflow* mostra-se interessante como um método de relação entre cooperação, composição e comunicação de serviços em ambientes de grade. Como (SPOONER et al., 2005) mencionam que o uso de *workflow* para construir aplicações grade oferece as seguintes vantagens:

- A possibilidade para construir e executar aplicações que podem coordenar recursos distribuídos em múltiplos domínios administrativos. Desta forma, obtendo capacidades de processamento específico, melhorando o *throughput* e reduzindo o custo de execução através da integração de múltiplas equipes envolvidas em partes diferentes do *workflow*, promovendo colaboração entre organizações;

Nas próximas seções serão abordados conceitos básicos, tipos, aspectos, linguagens e mo-

tores de execução utilizados para implementar *workflows* no ambiente de grade computacional.

3.1 Conceitos Básicos

Nesta subseção são explicados alguns conceitos básicos sobre a abordagem *workflow*, necessário para entendimento de alguns aspectos do trabalho de pesquisa apresentado nesta dissertação. Como descrito em (WFMC, 1999), são eles:

- **SGW**: Um sistema que define, cria e gerencia a execução de *workflows* através do uso de software, executando em um ou mais motores de execução, que é capaz de interpretar a definição do processo, interagir com participantes do workflow e também invocar o uso de ferramentas e aplicações de tecnologia da informação;

- **Motor de Execução**: um serviço de software ou motor que fornece funções operacionais para suportar a execução de instância de processo negociável, baseado em definições de processos. Estas funções incluem: interpretação de definição do processo, criação de instâncias de processo, gerenciamento de sua execução, incluindo começar, parar, suspender, reiniciar entre outros;

- **Atividade**: é a descrição de um pedaço de trabalho que forma um passo lógico dentro de um processo. Uma atividade pode ser manual ou automatizada por computador, pode também ser chamada de tarefa, nodo ou passo;

- **Processo de Negócio**: Um conjunto de um ou mais atividades ou procedimentos que coletivamente realizam um objetivo de negócio ou político, normalmente dentro do contexto de uma ou mais estruturas organizacionais definindo relacionamentos e regras funcionais. No ambiente de grade, o processo de negócio equivale ao problema complexo resolvido dentro das organizações virtuais que fazem parte da grade, *workflow* visa automatizar esse processo de negócio o máximo possível;

- **Definição do processo**: A representação de um processo de negócio em uma forma suporta manipulação automatizada, consistindo de uma rede de atividades e seus relaciona-

mentos, critérios para indicar o começo e o término do processo e informações sobre as atividades individuais, tais como participantes, associados aplicações de TI e dados;

- **Instância**: A representação única de um processo, ou atividade dentro de um processo, incluindo seus dados associados. Cada instância representa uma *thread* de execução separada do processo ou atividade, que pode ser controlada independentemente e ter seu próprio estado interno e identidade externamente visível;

- **Participante**: Um recurso que executa o trabalho apresentado por uma instância de atividade *workflow*. Podem-se identificar os participantes como humano, máquina, função e unidade organizacional;

- **WAPI**: é uma abreviação para *Workflow API and Interchange*, incorpora especificações para permitir interoperabilidade entre diferentes componentes de SGW e aplicações;

- **Aplicação**: Um termo geral para um programa que interage com serviço de representação, tratando parte ou todo o processamento requerido para suportar uma atividade particular. Possui duas categorias: aplicações clientes, que pedem facilidades e serviços de um motor de execução e aplicações invocadas, que suportam o processamento de atividades particulares que tem sua execução começada pelo motor de execução;

- **Dados Relevantes**: Dados que são usados por um motor de execução para determinar as transições de estado de uma instância *workflow* (por exemplo, pré e pós-condições e transições condicionais);

- **Dados de Controle**: Dados que apresentam o estado dinâmico do sistema *workflow* e suas instâncias de processo. Como por exemplo, o estado do processo ou atividade, que representa condições internas, definindo o estado de uma instância de processo ou atividade em um ponto particular no tempo (por exemplo, iniciado, executando, ativo, suspenso, completado, falhou ou indefinido);

- **Evento**: Uma ocorrência de uma condição particular (interna ou externa para o SGW), que induz o motor de execução a efetuar uma ou mais ações. Possui dois elementos:

Trigger, que lança um sinal que causa o começo de certa atividade ou notifica dados de controle do *workflow*; e a Ação que, é uma resposta ao sinal do *Trigger*;

- **Monitoramento**: A habilidade de rastrear e reportar os eventos *workflows* durante a execução do mesmo.

3.1.1 Estruturas de Definição

Esta subseção descreve a terminologia usada na definição e execução de processos para descrever a natureza do fluxo do processo e suas interações.

- **Roteamento Paralelo**: Um segmento de uma instância de processo, onde duas ou mais instâncias de atividade são executadas em paralelo;
- **Roteamento Seqüencial**: Um segmento de uma instância de processo, na qual várias atividades são executadas em seqüência sob uma única thread de execução;
- **AND-Split**: Um ponto onde uma única thread de controle divide em duas ou mais threads que são executadas em paralelo, permitindo que múltiplas atividades sejam executadas simultaneamente;
- **Condicional ou OR-Split**: Um ponto onde uma única thread de controle toma a decisão sobre qual ramificação pegar quando encontrar com múltiplas alternativas de ramificações;
- **Join ou AND-Join**: Um ponto onde duas ou mais atividades executando em paralelo convergem dentro de uma única thread de controle comum;
- **Iteração**: Um ciclo de atividade envolvendo a execução repetitiva de um ou mais atividades até uma condição ser encontrada;
- **Pré-Condição**: Uma expressão lógica que pode ser avaliada por um motor de execução para decidir se uma instância de processo ou atividade dentro de uma instância de processo pode começar a executar. Um AND-Join pode ser especificado na forma de

uma pré-condição, onde a condição requer que cada uma das threads convergentes tenha alcançado um status especifico;

- **Pós-Condição**: Uma expressão lógica que pode ser avaliada por um motor de execução para decidir se uma instância de processo ou atividade dentro de uma instância de processo tem sua execução terminada;

- **Transição**: Um ponto durante a execução de uma instância de processo onde uma atividade termina e a thread de controle começa a executar outra atividade, podendo ser condicional ou incondicional.

3.2 Tipos

Workflows são classificados de acordo com o nível de complexidade, flexibilidade, propósito e a estrutura das tarefas que fazem parte do *workflow*. Na literatura (ALONSO et al., 1997), (ALLEN, 2001), (WIKIPéDIA, 2006b) foram encontram 6 tipos de *workflow*, são eles:

- **Administrativo**: este tipo é caracterizado por não ser tão complexo, tratar rotinas administrativas e facilitar a definição do processo. Tipicamente, há muitas definições que funcionam simultaneamente e tendem a envolver um grande número de participantes, os passos a seguir são bem estabelecidos e existem uma grande quantidade de regras conhecidas por todos os envolvidos. As definições dos processos são geralmente feitas por formulário; se a definição for demasiadamente complexa em formulário então deve se usar outro produto. A flexibilidade é mais importante do que a produtividade nestes *workflows*. Exemplos desses tipos de *workflow* são: uma rotina de aprovação de débitos na mesma empresa, ou autorização de viagens;

- **Ad Hoc**: é aquele que necessita de intervenção humana de diferentes profissionais, não existe uma estrutura pré-definida para o processo, ou essa estrutura pode ser modificada em tempo de execução e os processos geralmente são repetitivos. Exemplos deste tipo de *workflow* são: documentação de produtos ou venda de produtos, onde não há um padrão

pré-determinado de movimentação de informação entre pessoas, por isso a necessidade de interação e coordenação humana;

- **Colaborativo**: é caracterizado principalmente pelo número dos participantes envolvidos e pelas interações entre eles. Ao contrário de outros tipos de *workflows*, que são baseados na premissa que há sempre um progresso avançado, este tipo pode envolver o excesso de diversas iterações na mesma etapa até que algum formulário do acordo seja alcançado ou até mesmo voltar à etapa anterior. Um exemplo bom deste tipo é a escrita de um artigo por diversos autores. Esse tipo é muito dinâmico, no sentido que é definido de acordo com seu progresso;

- **Transacional ou de Produção**: é caracterizado pela capacidade de gerenciar processos complexos que possuem uma estrutura fixa e um conjunto de regras de roteamento de mensagens entre as atividades ou tarefas. As aplicações de crédito, de empréstimo e as reivindicações de seguro são os exemplos típicos, mas nota-se que a diferença entre *workflows* administrativos e da produção é às vezes uma questão de ponto de vista. *Workflows* de produção levam em consideração na sua execução as seguintes características: ambientes de grande escala, complexos e heterogêneos, a variedade das pessoas e das organizações envolvidas no *workflow* e a natureza das tarefas. Geralmente, este tipo de *workflow* é utilizado em dois casos: primeiro caso, quando é necessário o monitoramento do status e a localização das tarefas e documentos manipulados pelo sistema em cada instante de tempo, o mecanismo de roteamento de mensagens é usado para notificação de status de cada tarefa, através de alertas ou avisos; segundo caso, quando utilizados em processos que ocorra pouca intervenção de pessoas, e quando estas intervenções ocorrerem, elas tem que ser simples e de pouca duração. Os *workflows* de grade computacional pertencem a esta categoria, pois os mesmos atuam em ambientes heterogêneos e em larga escala, exigem que processos complexos sejam os mais automatizados possíveis e visam alta produtividade ou desempenho;

- **Orientado a objetos**: é caracterizado por trazer os conceitos aplicados aos sistemas ori-

entados a objeto. Desta forma, envolvendo os conceitos de classe, herança e outras características onde os códigos e dados ficam em um mesmo elemento, o objeto. Sistemas *workflow* orientados a objetos são as versões mais sofisticadas dos sistemas de *Workflow* orientados a transações;

- **Baseado em Conhecimento**: é o mais recente e menos visto na literatura, utiliza técnicas de inteligência artificial, ou seja, o *workflow* aprende com acertos e erros.

3.3 Formalismos

O formalismo indica o relacionamento temporal entre as várias tarefas conectadas de acordo com suas dependências. A seguir serão apresentados os principais formalismos empregados na modelagem *workflow*. São eles:

- **Rede de Petri Colorida**: redes de Petri pertencem a uma classe especial de grafos orientados. Esse tipo de rede de Petri é bastante empregado em *workflow* de grade computacional (JENSEN, 1994). Ela possui os seguintes elementos gráficos: lugar, representado por círculos; transições, representadas por retângulos ou caixas, e arcos de lugares para transições e vice-versa, objetos perceptíveis e individuais que fluem através da rede como tokens, uma marcação inicial que define os tokens que cada lugar contém no inicio da execução e uma expressão para cada arco que representa um objeto individual. Redes de Petri descrevem execuções de tarefas seqüenciais, paralelas, *join*, *split* e *loops* e também execução condicional de tarefas, não permitindo somente modelagem do *workflow*, mas também seu controle da execução. A rede de Petri colorida modela a interação entre recursos de software representados por transições de software, e recursos de dados representados por lugares de dados. Se o *workflow* não for dependente do fluxo de dados, são empregados transições e lugares de controle que avaliam condições lógicas para determinar o fluxo de execução. O estado do *workflow* é representado pela marcação na rede de petri, na qual facilita a implementação de ferramentas para recuperação e *checkpoint* de *workflows* (HOHEISEL, 2005).

- **DAG e Não-DAG**: Um DAG (*Directed Acyclic Graph*) (AALST et al., 2000), (WIKIPéDIA, 2006a) é um grafo orientado com nenhum ciclo direcional, isto é, para qualquer vértice *v*, não existe um caminho que sai de *v* e chegue em *v* direcionalmente. Este formalismo permite representar estruturas como seqüência, paralelismos, join e split. DAG é largamente usado devido a sua estrutura simples. No entanto, não é possível definir ciclos ou *loops* entre as tarefas e descrever o estado das tarefas *workflows*, somente o comportamento. Além de todos os patrões contido em um workflow DAG, um workflow Não-DAG (YU; BUYYA, 2005) também inclui a estrutura de ciclos ou loops, na qual seções de tarefas *workflows* em bloco de iteração são permitidas. Esta estrutura é frequentemente usada em aplicações científicas. Este formalismo está posicionado entre DAG e rede de Petri, ou seja, permite representar loops, mas não apresenta a quantidade de elementos gráficos e a complexidade das redes de petri;

3.4 Especificação ou Modelo

Define um *workflow*, incluindo suas definições de tarefas e estruturas (YU; BUYYA, 2005). Existem dois tipos de modelos *workflows*, que são:

- **Modelo Abstrato**: o *workflow* é descrito em uma forma abstrata na qual o *workflow* não tem especificado recursos da grade para execução de tarefas, fornecendo um modo flexível para usuários definirem *workflow* sem se preocuparem com detalhes de implementação de baixo nível;

- **Modelo Concreto**: especifica tarefas *workflows* em recursos da grade (por exemplo, transferência de dados de entrada e saída das tarefas).

3.5 Motores de Execução

Motor de Execução é um ambiente que ajuda ou executa a coordenação e controle das tarefas *workflows* que estão especificadas no modelo abstrato ou concreto, sendo um componente

fundamental de um sistema de execução de *workflows*. Dentre suas funções estão: interpretar processos definidos em uma linguagem *workflow* específica, criar e gerenciar execuções de instâncias de processo, invocar programas e serviços de grades para o processamento de tarefas ou atividades *workflow*, negociar o fluxo de mensagens entre as tarefas, mapear *workflows* abstratos em *workflows* concretos ou executáveis entre outras funcionalidades (WFMC, 1999). Existem vários motores *workflows* desenvolvidos para diversos sistemas *workflows* em ambientes de grade computacional. Desta forma, nesta seção serão apresentados os principais. São eles:

- **DAGMan** (TANNENBAUM et al., 2001) é um meta-escalonador para tarefas Condor que trata as dependências entre as tarefas. Usa DAG como estrutura de dados para representar dependências entre tarefas, sendo que cada tarefa é um nodo no grafo e os arcos identificam suas dependências. Tarefas filhos só podem executar quando as tarefas pais tiverem seu processamento terminado. Ciclos ou iterações entre tarefas não são permitidos. Usuários precisam especificar transferências de dados através de comandos pré-processados ou pós-processados associados com o processamento da tarefa. A execução das tarefas individuais é gerenciada pelo escalonador Condor. Assim se a tarefa falha devido à natureza do sistema distribuído, ela será restabelecida pelo Condor enquanto DAGMan não está consciente destas falhas. Entretanto, DAGMan é responsável por relatar erros do conjunto de tarefas submetidos e gerar um histórico ou log da estrutura DAG destas tarefas submetidas. No caso de uma tarefa falhar, o restante do DAG continua até nenhum progresso ser alcançado. Uma tarefa com falha pode tentar novamente um número configurável de vezes. O histórico DAG indica as porções incompletas do DAG. Assim, usuários podem corrigir os erros de tarefas que falharam e re-submeter o DAG destas tarefas;

- **GridAnt** (AMIN et al., 2004) foi desenvolvido pelo *Argonne National Laboratory*, tendo sido projetado para usuários do *middleware* Globus (GLOBUS, 2006) versão 2 e 3 com pouca experiência em sistemas complexos de *workflows*, é uma ferramenta para expressar e controlar a seqüência de execução das tarefas. O motor GridAnt é o controlador

central que trata dependências entre as tarefas, recuperação de falhas, análise de desempenho, sincronização do processo e comunicação entre tarefas internas. É uma extensão da ferramenta de construção de processos Java, chamada Ant (APACHE..., 2006). Este motor de execução suporta construtores como constantes, expressões aritméticas, variáveis globais e outras estruturas em tempo de execução. Tags são usadas pelos usuários para predefinir as tarefas grades e construir *workflows* complexos; usuários também podem monitorar o progresso de execução por meio da ferramenta de visualização gráfica desenvolvida neste projeto;

- **GWFE** foi desenvolvido pelo projeto Gridbus (BUYYA; VENUGOPAL, 2004) e permite ao usuário conectar aplicações e executar *workflows* em ambientes de grade. Fornece uma linguagem *workflow* baseada em XML para usuários definirem tarefas e suas dependências. Este motor de execução fornece uma arquitetura de escalonamento hierárquico para adaptar ambientes dinâmicos e heterogêneos como ambiente de grade, escalonando tarefas orientadas a eventos usando modelo *tuple-space*. Um mecanismo orientado a evento com uma abordagem de notificação torna a execução flexível e fracamente acoplada. Permite que a decisão de escalonamento seja feita em tempo de execução da tarefa e que as tarefas que falharam na execução sejam re-escalonadas em um recurso alternativo (YU; BUYYA, 2005);

- **Karajan** (LASZEWSKI; HATEGAN, 2005) é uma linguagem de especificação *workflow* e motor de execução baseado no Globus Toolkit (GLOBUS, 2006), sendo parte do *Java CoG Kit* (LASZEWSKI et al., 2001) e derivada do projeto GridAnt (AMIN et al., 2004). O *Java CoG Kit* é baseado em um projeto modular e fornece mecanismos para desenvolvimento rápido de aplicações e fácil integração com variedades de versões do Globus e também Condor. Além disso, Karajan permite utilizar abstrações de programação de execuções de tarefas e transferências de arquivos do Java CoG kit. Além da estruturas de paralelismo e seqüência permitidas em quase todas as linguagens *workflows*, esta linguagem também suporta as seguintes estruturas de instruções de controle de execução: *join*, *split*, iterações, operadores booleanos e matemáticos e estruturas de dados avançadas como

list, *range* e tabelas *hash* para tarefas repetitivas. O modelo de execução inclui elementos de execução que podem ser as tarefas *workflows* a serem executadas e também eventos gerados por estes elementos durante a execução *workflow* ou pelo ambiente grade, que podem causar diferentes ações. Elementos de execução podem estar em diferentes estados, dependendo do corrente status de execução. Escalonamento dinâmico de elementos é feito de um modo simples, ou seja, uma lista de recursos disponíveis. Vários métodos de tratamentos de falhas são suportados, permite ao usuário integrar estratégias de erros e exceções dentro do *workflow*. *Checkpointing* permite ao usuário armazenar estados intermediários da execução do *workflow* para mais tarde voltar ao estado correto quando ocorrem problemas na execução de determinado elemento;

- **Pegasus** (DEELMAN et al., 2004) foi desenvolvido no projeto GriPhyN (DEELMAN et al., 2002). Este motor de execução faz o mapeamento de *workflows* abstratos em *workflow* concretos. Um *workflow* abstrato pode ser construído por consulta em Chimera (FOSTER et al., 2002) ou fornecido por um usuário em formato DAX (descrição XML de DAG). O *workflow* abstrato descreve computação em termos de arquivos e aplicações lógicas, indicando suas dependências na forma de um DAG. Pegasus reduz o *workflow* abstrato pelo reuso de conjunto de dados materializados que foram produzidos por outros usuários da organização virtual. Neste motor assume-se que é mais caro produzir novos conjuntos de dados do que acessar dados já produzidos anteriormente. Desta forma, o algoritmo de redução remove quaisquer antecedentes de tarefas redundantes que não tem qualquer descendente desmaterializado com o objetivo de reduzir a complexidade dos *workflows* concretos. Pegasus usa RLS (FOSTER, 2005) para localizar réplicas de dados, MDS (FOSTER, 2005) para localizar recursos e obter suas características e TC (DEELMAN et al., 2002) para localizar os componentes de aplicações. Ele também é capaz de gerar um *workflow* automaticamente baseando em uma descrição de metadados do produto de dados requisitados com ajuda de técnicas de planejamento de inteligência artificial. Os *workflows* concretos gerados por este motor são transformados em tarefas Condor com gerenciamento de execução através do Condor DAGMan (TANNENBAUM et al., 2001);

- **FreeFluo** é uma ferramenta de orquestração de *workflows* que trata invocações Web Services baseada em WSDL. Inicialmente desenvolvida pela *IT Innovation* (INNOVATION, 2006) e atualmente faz parte como componente do sistema Taverna (OINN et al., 2004) no projeto myGrid (STEVENS; ROBINSON; GOBLE, 2003). Taverna tem como propósito ajudar biólogos a desenvolverem e executarem *workflows* de bioinformática em grade computacional. FreeFluo (MYGRID, 2006) é integrado dentro da Taverna como motor de execução para transferir dados intermediários e invocar serviços. Este motor suporta as linguagens *workflows* WSFL e SCUFL/XScufl e é muito flexível, ou seja, em seu núcleo está um *framework* de orquestração reusável que se prende à arquitetura de execução. FreeFluo também inclui extensões de bibliotecas que permitem a execução de *workflows* escritos em um subconjunto de WSFL. Existe suporte para a descoberta através do padrão UDDI (CURBERA et al., 2002) e registro de procedência. FreeFluo aceita dados de entrada e arquivos de definição do Taverna e coordena o escalonamento dos serviços *workflows*. Frequentemente, em análises de bioinformática, os dados de saída de um serviço são entrada do próximo serviço, se dois serviços não tem influência um sobre outro, este motor pode programar estes eventos para acompanhar um ao outro; assim o motor determina a representação destes serviços ao mesmo tempo, reduzindo o tempo gasto para executar todo o *workflow*. Quando um serviço é invocado pelo motor ou dados de saída finais ou intermediários são produzidos, um novo evento é gerado para anunciar esta mudança. O progresso do *workflow* pode ser monitorado usando uma interface de visualização do Taverna. Os serviços podem possuir os seguintes status: programado, invocado, completado, indisponível e com falhas;

- **Grid Job Handler** é o motor de execução do projeto *Fraunhofer Resource Grid* (FRAUNHOFER..., 2006). Além da execução de simples e não acopladas aplicações, nas quais tem sido definida pela linguagem de descrição específica do Globus chamada XML-RSL (GLOBUS, 2006), Grid Job Handler (HOHEISEL; DER, 2003) permite a execução de tarefas de grade acopladas, baseado na linguagem de descrição GJobDL, onde fluxo de controle e dados são descritos por uma rede de Petri (JENSEN, 1994). Este motor usa o serviço

GRAM (FOSTER, 2005) com objetivo de executar os componentes únicos de aplicações em nodos da grade. Também utiliza o serviço GridFTP (FOSTER, 2005) para transmitir os dados necessários para arquivos executáveis no computador alvo antes da execução da tarefas.

3.6 Comparativo entre os Motores de Execução

Para possibilitar uma melhor compreensão a respeito dos motores de execução descritos na seção anterior, nesta seção será apresentada duas tabelas, contendo uma comparação geral entre os motores de acordo com certas características. Na Tabela 1 consta uma comparação com características gerais dos motores de execução. A Tabela 2[1] descreve as características de definição *workflow* utilizados por esses motores de execução.

Analisando a Tabela 1, é possível perceber que o motor de execução Karajan mostrou-se o mais versátil dentre todos os descritos, permitindo um número variado de opções de interface, interação com várias versões do Globus Toolkit e também com o Condor e com qualquer tipo de aplicação grade.

Já na Tabela 2 podemos notar novamente que o motor de execução Karajan é mais versátil se tratando da definição e composição de *workflow*. As características que se destacam são: permite todas as estruturas de definição de *workflow* fornecidas também em outras linguagens, estruturas de definição avançadas (por exemplo, *list*, *range* e *hash table*) que não estão presente em outras linguagens e motores de execução, e podem auxiliar na definição de *workflows* complexos e composição supervisionada pelo usuário através de uma GUI ou pela própria linguagem suportada por este motor de execução.

[1]O termo "N"denota que não está presente aquela característica ou não foi especificada e a abreviação "S. U."significa supervisionado pelo usuário

Tabela 1: Resumo dos Motores de Execução

Motores	Interface	Pré-requisitos	Grade	Aplicações	Licença
DAGMan	Linha de Comando	Condor	GT2 (Condor_G)	Sob-Demanda	GPL
GridAnt	GUI	Java SDK 1.4 e CoG	GT2, GT3 eGT4	Sob-Demanda e Alto Desempenho	GTPL
GWFE	Linha de Comando	GT2, mysql, IBM tuples space, CoG	GT2 e GT3	Grande Quantidade de dados	GPL
Pegasus	N	Condor, DAGMan, RLS	Globus e Condor	Grande Quantidade de dados	GTPL
Grid Job Handler	GUI	Globus	GT2 e GT3	Sob-Demanda	GTPL
FreeFluo	GUI	Java SDK 1.4	N	Grande Quantidade de dados	LGPL
Karajan	Linha de comando, GUI e API	Java SDK 1.4 e CoG 4.1.X	GT2, GT3, Condor e GT4	Quaisquer aplicações que necessitem acessar a grade	GTPL

Tabela 2: Características de definição workflow nos Motores de Execução

Motores	Form.	Estruturas	Modelo	Ling.	Composição
DAGMan	DAG	Seqüenciais, paralelas, join e split	Abstrato	N	S. U.: linguagem
GridAnt	Não DAG	Seqüenciais, paralelas, join, split e loops	Concreto	Ant-like	S. U.: linguagem
GWFE	DAG	Seqüenciais, paralelas e parâmetros	Abstrato/ Concreto	Baseada em XML	S. U.: linguagem
Pegasus	DAG	Seqüenciais, paralelas, join e split	Abstrato	Baseada em XML	Automático
Grid Job Handler	Redes de Petri	Seqüenciais, paralelas, join, split e loops	Abstrato/ Concreto	GJobDL	S. U.: GUI
FreeFluo	DAG	Seqüenciais, paralelas, join e split	Abstrato/ Concreto	WSFL e SCUFL/ XScufl	S. U.:linguagem e GUI
Karajan	Não DAG	Seqüenciais, paralelas, join, split, loops, list, range e hash table	Abstrato	Karajan	S. U.: linguagem e GUI

4 *Grades Computacionais e Dispositivos Móveis*

Computação móvel é um paradigma computacional que um dos objetivos é fornecer ao usuário acesso a infra-estrutura de redes sem fio sem levar em consideração sua localização física. Sendo assim, fornecendo ao usuário informações, aplicações, recursos e serviços em qualquer hora e lugar através de dispositivos móveis como PDAs, telefones celulares, laptops e outros (FORMAN; ZAHORJAN, 1994). Como pode ser visto em (ANATEL, 2006), tem sido verificado um aumento no número de dispositivos móveis no mercado com o passar dos anos. Isto mostra que cada vez mais, um número maior de pessoas está adquirindo informações, utilizando serviços e executando aplicações em qualquer hora e lugar através de seus dispositivos móveis.

Apesar dos dispositivos móveis oferecerem recursos interessantes como, por exemplo, receptores GPS, câmeras, microfones e principalmente sensores (MCKNIGHT; HOWISON; BRADNER, 2004), existem aplicações (como por exemplo, seqüenciamento do genoma, cosmologia, análise de imagens espaciais e outras) que requisitam recursos que forneçam uma grande quantidade de recursos, armazenamento ou processamento intensivo de dados, e/ou alta taxa de transmissão de dados, recursos que não estão disponíveis em dispositivos móveis, devido às séries de restrições de recursos encontrados nestes dispositivos. As principais restrições que dificultam a execução destas aplicações são apresentadas abaixo:

- Instabilidade da rede sem fio, uma vez que este tipo de rede está sujeita a interferência de fatores externos do meio ambiente ou atmosférico (por exemplo, chuva e/ou tempestade eletromagnética) em seus canais de comunicação;

- Baixa largura de banda da rede sem fio;
- Pouca capacidade de armazenamento de dados;
- Baixo poder de processamento da CPU;
- Pequenos dispositivos de entrada e saída (por exemplo, tela e teclado);
- Dependência do tempo de vida da bateria.

No entanto, o dispositivo móvel pode interagir com a grade computacional, onde o ambiente de grade pode oferecer o compartilhamento de recursos em larga escala aos dispositivos móveis para processar as aplicações que ajudam a resolver problemas complexos, fornecendo assim maior flexibilidade, desempenho e confiabilidade para o usuário de dispositivos móveis. Desta forma, os usuários destes pequenos aparelhos podem requisitar a execução de vários tipos de aplicações sem se preocupar com limitações de recursos impostas pelos mesmos, já que eles estão utilizando os recursos compartilhados na configuração da grade.

Sob esta ótica, várias pesquisas (CLARKE; HUMPHREY, 2002), (CHU; HUMPHREY, 2004), (GONZALEZ-CASTANO; VALES-ALONSO; LIVNY, 2003), (PHAM; HUANG; DULAN, 2002), (LITKE; SKOUTAS; VARVARIGOU, 2004), (AHMAD et al., 2004), (MCKNIGHT; HOWISON; BRADNER, 2004) propõem dispositivos móveis acessando os recursos compartilhados na configuração da grade e também fazendo parte desta configuração de grade. Em (PARK; KO; KIM, 2003) são apresentados dois modos de interação dos dispositivos móveis com o ambiente de grade, que são descritas logo abaixo:

- **Interface**: neste ambiente o dispositivo móvel age somente como uma interface, permitindo submissão, monitoração e gerenciamento de tarefas em qualquer lugar e hora. A infra-estrutura da configuração de grade representa mais desempenho e confiabilidade para aplicações submetidas dos equipamentos sem fio ao ambiente. Em outras palavras, o equipamento móvel é usado como uma entrada para a configuração de grade;

- **Recurso**: nesta segunda classificação os dispositivos móveis são considerados como parte da própria grade. Assim, laptops e PDAs fornecem requisitos computacionais quando conectados em hotspots, formando uma configuração de grade.

A interação dos dispositivos móveis com os ambientes de grade é recente, sendo os trabalhos nesta área são, em sua maioria, posteriores ao ano de 2002. Apesar de estar havendo uma melhora significante no poder de processamento e taxa de transmissão de dados nos dispositivos móveis, assim como mostrado por (SAIRAMESH et al., 2004), algumas aplicações, como exemplificadas anteriormente, exigem uma quantidade de recursos que excede a atual capacidade de processamento e armazenamento destes dispositivos.

Em outras palavras, a capacidade computacional destes dispositivos ainda é muito pequena, comparada com os aglomerados de computadores e grades computacionais. Seguindo esta visão, nesta dissertação foi desenvolvida uma abordagem empregando o modo de interação Interface, onde os usuários destes equipamentos podem usufruir do grande poder computacional oferecido pelas configurações de grade. A seguir, serão mostrados alguns trabalhos envolvendo a interação entre dispositivos móveis e o ambiente de grade empregando o modo de interação Interface.

4.1 Abordagem Ahmad, Ikram, Ali, Anjum, Steenberg, Newman, Bunn, Thomas e Azim

No artigo (AHMAD et al., 2004) apresentam o projeto e implementação de um ambiente de análise interativa em grades computacionais para PDAs. O objetivo deste trabalho de pesquisa foi desenvolver uma conjunto de aplicações de análises físicas otimizadas ao máximo para obter o melhor desempenho utilizando PDAs.

Desta forma, foi desenvolvido uma versão Java do framework *Clarens*, chamado *JClarens*. Este framework baseado em *Web Services* e grade contém serviços como autenticação baseada em segurança (GSI), acesso a dados remotos, submissão e monitoração de status de tarefas. Os clientes, que neste caso são dispositivos móveis como PDAs, comunicam-se com o *JClarens*

usando o XML-RPC. Uma vez conectados ao *JClarens*, os clientes podem acessar os serviços de grade fornecidos pelo portal.

O sistema de escalonamento de tarefas Condor (THAIN; TANNENBAUM; LIVNY, 2005) é responsável pelo escalonamento de tarefas na arquitetura do *JClarens* que fornece uma interface para clientes submeterem tarefas ao Condor. O portal *JClarens* também permite que dispositivos móveis monitorem o status das tarefas em um ambiente Condor e possam visualizar o resultado da execução. Levando em consideração que as tarefas submetidas podem consumir bastante tempo e também um grande número de arquivos de entrada, usuários de dispositivos móveis podem trabalhar por trás de uma simples interface, enquanto estas tarefas complexas são executadas e gerenciadas pelo *JClarens* em uma configuração de grade.

Para clientes conectados a uma rede estruturada existem várias aplicações de grade disponíveis para análise física, tais como JAS, Root, PAW e Wired. Desta forma, na abordagem proposta por (AHMAD et al., 2004) foram selecionadas as aplicações JAS e Wired para serem conduzidas e redefinidas para dispositivos móveis. Por fim, foi realizada uma avaliação comparativa de desempenho do ambiente, onde havia um PDA (iPAQ h3955) e dois computadores desktops com processadores Pentium II e IV.

4.2 Abordagem de Hwang e Aravamudhan

Em (HWANG; ARAVAMUDHAM, 2004) é desenvolvida a proposta de uma arquitetura de *middleware* escalável chamada *Scalable Inter-Grid Network Adaptation Layers* (Signal), integrando dispositivos móveis com configurações de grade, permitindo que estes dispositivos possam submeter tarefas ao ambiente de grade através de sistemas baseado em *proxy*. Diferentes dispositivos móveis podem interagir com uma infra-estrutura de grade através do sistema *proxy* pela troca de vários parâmetros tais como status e especificação de QoS.

Esta arquitetura negocia a adaptação de rede e QoS entre a infra-estrutura de rede e os dispositivos móveis. A adaptação de rede descreve como a conectividade e disponibilidade de recursos de rede deve ser coordenada entre redes de grade e redes sem fio. Já a adaptação de

QoS é o gerenciamento de QoS entre diferentes recursos através do sistema *proxy*. As principais atividades realizadas pelo sistema ou servidor *proxy* são:

- Interagir com os alguns serviços fornecidos pelo Globus Toolkit, como por exemplo, MDS (FOSTER, 2005), para comunicar a disponibilidade de recursos e MMJFS e GRAM (FOSTER, 2005), os quais fornecem um escalonador de tarefas em tempo real;
- Ser capaz de considerar a ordem nas quais tarefas são submetidas e quais nodos são disponíveis para executar estas tarefas, além de manter a QoS requerida durante algumas mudanças de ambiente;
- Monitorar nodos móveis, fornecendo medidas e relatando entrega de QoS atual;
- Realizar a reserva de recursos necessária quando recebe um pedido de submissão.

Para enviar um pedido de submissão de tarefa para o ambiente de grade, o dispositivo móvel primeiramente autentica-se no servidor *proxy* usando o serviço GSS e passa o código do pedido usando o protocolo SOAP. Logo em seguida, o servidor *proxy* analisa o código e checa a alocação e disponibilidade de recurso, enviando o pedido de submissão de tarefa para um recurso em localização remota da grade (por exemplo, outros dispositivos móveis, aglomerados de computadores e máquinas *desktop*), retornando um relatório de status da execução para o dispositivo móvel e também retornando o resultado no término da execução. Devido à execução remota, os dispositivos móveis consomem muito pouca energia e usam a largura de banda efetivamente.

Nesta arquitetura também ocorre o emprego de vários parâmetros, tais como *caching* e *prefetching* de dados para aplicações melhorarem desempenho, disponibilidade e QoS. Por causa do limite de tráfico de controle necessário para encontrar um dispositivo móvel, tal *caching* melhora o desempenho pelo decréscimo de tempo necessário para o servidor *proxy* localizar o dispositivo.

4.3 Abordagem de Chu e Humphrey

O artigo (CHU; HUMPHREY, 2004) descreve o projeto, implementação e avaliação de um *framework* chamado Mobile OGSI.NET. Este *framework* foi criado para promover o compartilhamento de recursos e colaboração entre dispositivos móveis que melhore o acesso por parte do usuário móvel ao ambiente de grade computacional. Mobile OGSI.NET estende uma implementação de grade computacional chamada OGSI.NET. Também discute as limitações de recursos e a intermitência da rede sem fio em dispositivos móveis. Os objetivos deste *framework* são:

- Construir uma plataforma que forneça o melhor potencial para colaboração entre um ou mais dispositivos móveis na resolução de um problema;
- Suportar a colaboração entre dispositivos móveis e computadores servidores ou *desktop*, abrindo a possibilidade de maximizar a utilização de recursos com menos limitação;
- Operar em um número grande de dispositivos móveis. Sendo assim, este framework foi desenvolvido em um sistema operacional largamente usado;
- Discutir a limitação dos recursos dos dispositivos móveis com o intuito de oferecer um uso destes recursos de forma otimizada.

Alguns aspectos motivaram o desenvolvimento do Mobile OGSI.NET, tais como: familiaridade com a especificação OGSI através da implementação do projeto OGSI.NET e também o fato de nem OGSI.NET e nem Globus Toolkit executarem em dispositivos móveis.

Mobile OGSI.NET foi desenvolvido para o sistema operacional da família Microsoft PocketPC. Desta forma, este framework foi implementado no topo do framework .NET Compact, agindo como uma camada de software intermediária entre a aplicação e o sistema operacional. Além disso, esta camada oferece uma GUI e também utiliza várias aplicações e bibliotecas desenvolvidas no framework .NET.

O Mobile OGSI.NET acomoda dispositivos que estão rapidamente desligando-se, requisitando assim as capacidades de migração de processos e execução distribuídas. O objetivo em longo prazo é silenciosamente migrar processos quando notado que a carga da bateria está se tornando perigosamente baixa.

4.4 Abordagem de González-Castaño, Vales-Alonso e Livny

A tradicional interface do Condor (THAIN; TANNENBAUM; LIVNY, 2005) é orientada a linha de comando. Entretanto, os autores optaram por projetar *front-ends* específicos para dispositivos móveis. Desta forma, para alcançar este objetivo (GONZALEZ-CASTANO; VALES-ALONSO; LIVNY, 2003) apresentaram uma proposta de interface em dispositivos móveis para acessar o poder computacional oferecido pelos computadores que fazem parte de uma configuração Condor. Em outras palavras, procurou-se desenvolver interfaces melhoradas e adaptadas para serem utilizadas em navegadores de internet instalados nos dispositivos móveis.

Desta forma, adotou-se o protocolo HTTP para a arquitetura proposta e os navegadores dos clientes móveis que suportam: WML, XML, HTML, c-HTML e outros mais. Ao invés de focar em um dispositivo específico, foi definida uma hierarquia de interações de usuários de acordo com as características específicas de hardware de cada dispositivo. Assim, estas características de hardware dividiram os dispositivos em três categorias ou camadas hierárquicas, como pode ser notado na pirâmide da Figura 3. Uma vez que as tecnologias e características das diferentes camadas hierárquicas foram determinadas, ocorreu o desenvolvimento de *front-ends* Condor para as duas camadas mais superiores: PDA e telefones celulares WAP.

Na camada PDA as seguintes funções são disponibilizadas: checa status da fila; remove tarefas da fila, submete novas tarefas; e edita parcialmente código fonte. Na camada celular são suportadas quase todas as características da camada PDA, com exceção da capacidade de edição parcial de código fonte.

Figura 3: Camadas Hieráquicas de Dispositivos (GONZALEZ-CASTANO; VALES-ALONSO; LIVNY, 2003).

4.5 Abordagem de Brooke e Parkin

A abordagem proposta por (BROOKE; PARKIN, 2005) apresenta a implementação de um cliente PDA de grade computacional que permite segurança na submissão de tarefas síncronas e assíncronas, monitoração de tarefas, entrega de resultados destas tarefas e uma facilidade para *upload* e *download* de arquivos em um PDA, usando a infra-estrutura de rede *wireless* e o *middleware* UNICORE (ERWIN; SNELLING, 2001).

A arquitetura UNICORE obrigatoriamente exige que a comunicação entre o cliente e o *middleware* seja feita sobre o protocolo SSL. Desta forma, um suporte a segurança foi implementado para que o cliente PDA pudesse submeter tarefas a este *middleware*. O aspecto de suporte a segurança no cliente PDA é o principal foco desta abordagem e foi crucial para a escolha das ferramentas utilizadas no desenvolvimento do cliente PDA. Primeiramente tentou-se desenvolver um cliente utilizando Personal Java (MICROSYSTEMS, 2006) em um dispositivo PDA: Sharp Zaurus SL-5500. No entanto, ocorreram problemas na tentativa de abrir o *PKCS12 keystore* na implementação do cliente PDA com Personal Java, dificultando o estabelecimento de uma conexão segura entre o PDA e o *middleware* UNICORE. Sendo assim, foi adotado o

J2ME (Sun Microsystems, 2006b) como ambiente de execução. A implementação da segurança especificada pelo UNICORE foi alcançada com sucesso utilizando J2ME, garantindo uma comunicação segura entre cliente PDA e o *middleware* UNICORE.

Desta forma, os próximos passos foram fornecer uma interface ao usuário que permitisse preparar tarefas, submeter tarefas e visualizar os resultados obtidos na execução das tarefas.

4.6 Considerações sobre as Abordagens Propostas

Como pode ser visto nas seções 4.1, 4.2, 4.3, 4.4, 4.5, todas as cinco abordagens implementadas permitem submissão de tarefas na configuração da grade a partir do dispositivo móvel, principalmente PDAs. No entanto, a maioria destes trabalhos, incluindo (AHMAD et al., 2004), (HWANG; ARAVAMUDHAM, 2004), (GONZALEZ-CASTANO; VALES-ALONSO; LIVNY, 2003) e (CHU; HUMPHREY, 2004), não trata a submissão de várias tarefas de forma controlada e automatizada para a resolução de um único problema, sendo que as tarefas são submetidas e monitoradas uma por vez na interface do dispositivo e tendo a ordem de submissão deste conjunto de tarefas controlada pelo usuário.

Além disso, algumas abordagens possuem interfaces poucos amigáveis (por exemplo (GONZALEZ-CASTANO; VALES-ALONSO; LIVNY, 2003)), o usuário é obrigado a conhecer comandos específicos do sistema Condor para submeter e monitorar tarefas a partir do dispositivo móvel, responsável por gerenciar as tarefas submetidas ao ambiente de grade. Isto, torna mais complicado o processo de submissão e monitoração para usuários com pouca experiência em grades computacionais. Já em (CHU; HUMPHREY, 2004), a abordagem proposta é suportada somente em dispositivos móveis com o sistema operacional Microsoft PocketPC 2003 e não permite monitoração de status das tarefas submetidas.

Assim como (GONZALEZ-CASTANO; VALES-ALONSO; LIVNY, 2003) destaca nos trabalhos futuros deste artigo, é necessário fornecer interfaces melhoradas que escondam detalhes dos mecanismos e políticas de cada *middleware* de grade, como por exemplo balanceamento de carga e políticas de segurança, e também possibilitem a submissão e monitoração de várias

tarefas de uma forma controlada, combinada e automatizada, permitindo que os usuários destes dispositivos possam melhor usufruir os recursos compartilhados pela configuração de grade.

Na abordagem apresentado por (BROOKE; PARKIN, 2005), é comentado que o cliente PDA pode montar um fluxo de execução de tarefas (isto é, um *workflow*) com os necessários AJOs (isto é, objetos de tarefas abstratas UNICORE), onde AJO é uma biblioteca Java para a modelagem de tarefas UNICORE (mais detalhes em (UNICORE…, 2006)). No entanto, (BROOKE; PARKIN, 2005) revelam que no cliente PDA somente está implementado um pequeno conjunto de funcionalidades AJOs, não sendo claramente especificadas quais funcionalidades são implementadas, os autores comentam também que poderiam mais adiante estender e melhorar a combinação de AJOs dentro de *workflows* mais complexos e especializados. Outro ponto fraco deste artigo foi o fato dos autores não mencionarem nenhuma readaptação, otimização ou personalização nas interfaces, com intuito de melhor adequar esta abordagem ao conjunto de limitações encontradas nestes dispositivos.

Além disso, não foram especificadas as estruturas de fluxo de execução de tarefas e operações no sistema de tarefas do AJOs que são suportadas e também não foi mostrado qualquer estudo de caso, teste de funcionalidade ou análise comparativa que evidenciasse a composição do fluxo de tarefas. Portanto, apresentando-se poucos detalhes da interface do cliente PDA com relação a composição, submissão e monitoração das tarefas *workflows* (por exemplo, não especifica se o status de execução fornecido é do *workflow* como um todo ou de cada tarefa que faz parte do *workflow*).

O principal foco do artigo (BROOKE; PARKIN, 2005) é a implementação de segurança nos clientes PDAs. Os autores destacam que somente utilizando uma comunicação segura é possível submeter tarefas no *middleware* UNICORE, ou seja, comunicação segura é um requisito obrigatório deste *middleware* grade. Desta forma, é possível identificar que a funcionalidade *workflow* em ambientes de grade a partir do PDA fornecida por esta proposta é reduzida e também não especificada claramente.

A tabela 3 mostra as características dos trabalhos relacionados de forma resumida.

Tabela 3: Resumo das Características dos Trabalhos Relacionados

Trabalhos	Suporta Celulares	Middleware Grade	Suporta Workflow	Interfaces Adaptadas
(AHMAD et al., 2004)	Não	Condor	Não	Sim
(HWANG; ARAVA-MUDHAM, 2004)	Não	Globus	Não	Não
(CHU; HUMPH-REY, 2004)	Não	OGSI.NET	Não	Não
(GONZALEZ-CASTANO; VALES-ALONSO; LIVNY, 2003)	Sim	Condor	Não	Sim
(BROOKE; PARKIN, 2005)	Não	Unicore	Sim	Não

5 *Abordagem proposta e Resultados Experimentais*

Conforme apresentado no capítulo 4, os dispositivos móveis enfrentam dificuldades com relação à submissão e monitoração de várias tarefas a uma configuração de grade que trabalham cooperativamente para resolver um único problema. A maioria dos esforços de pesquisa desenvolvidos até o presente momento não apresentam uma forma organizada, combinada, automatizada e coordenada para a submissão e monitoração dessas várias tarefas. No entanto, o crescente volume e distribuição de dados e tarefas em diversas áreas de pesquisa torna cada vez mais complexos as descobertas e analíse de novas informações para a resolução de um único problema.

Além disso, como são inúmeras as tarefas que podem ser executadas e também suas respectivas interdependências, é necessário um mecanismo que auxilie na organização, coordenação e combinação destas diversas tarefas juntamente com a grande quantidade de dados distribuídos em diversas organizações virtuais da configuração de grade. Portanto, um mecanismo com estas características pode fornecer uma forma mais elaborada para resolver problemas complexos através da utilização de dispositivos móveis, enquanto os pesquisadores e/ou profissionais estão se deslocando por diferentes lugares. Desta forma, a presente dissertação propõe como alternativa como solução para melhor submeter e monitorar várias tarefas executando em configurações de grade através da utilização de *workflows*. Estes são utilizados, primeiramente, para organizar, controlar, combinar a execução de múltiplas tarefas que trabalham cooperativamente para resolver um único problema.

A partir do uso de *workflows*, não existe a necessidade de preocupação com o controle e or-

denação do fluxo de execução das tarefas por parte do usuário do dispositivo móvel. Além disso, similarmente aos trabalhos de pesquisa apresentados em (SHAHAB et al., 2004; LASZEWSKI; HATEGAN, 2005), nesta dissertação também foi considerado importante a monitoração do *workflow*, uma vez que algumas execuções de *workflows* podem levar horas ou até dias para serem concluídas (OINN et al., 2004). Uma vez que estes aparelhos permitem uma maior portabilidade do que computadores desktops, por serem mais leves, menores e permitirem aos seus usuário se conectarem a redes estruturadas sem a necessidade de estar conectados por cabo, estes possibilitam que o usuário tenha conhecimento do status e resultados da execução dos *workflows* submetidos à configuração de grade de diversos lugares.

Desta forma, foi desenvolvida uma interface única e padronizada que fornece informações aos usuários sobre a execução de cada tarefa do *workflow* (por exemplo, status e tempo de execução de cada tarefa), permitindo que o usuário possa averiguar se uma tarefa está executando como esperado e também detectar falhas ou problemas na execução de alguma tarefa. Além disso, não exige que o usuário conheça os comandos específicos do *middleware* de grade e oculta certos detalhes da submissão e monitoração das tarefas *workflow*. Exemplos desses detalhes são: políticas de balanceamento de carga e segurança; localização de onde as tarefas estão sendo executadas; e quais serviços de grade estão sendo utilizados para a execução das mesmas.

Este capítulo apresenta o ambiente experimental desenvolvido para validar a abordagem proposta. O objetivo deste ambiente é possibilitar a submissão e monitoração de *workflows* em configurações de grade utilizando diferentes dispositivos móveis. Seu desenvolvimento iniciou-se com o estudo das possibilidades de implementação. Desta forma, foi necessário identificar as características principais que este deveria apresentar, para que então, fossem selecionadas as opções mais adequadas às características dos dispositivos móveis adotados para a implementação da abordagem. A descrição do ambiente experimental, assim como as justificativas de algumas das opções de implementação adotadas neste ambiente, são discutidas na seção 5.1. A seção 5.2 apresenta a arquitetura desenvolvida. Na seção 5.3 é apresentado o protótipo implementado. Na seção 5.4 é mostrado uma análise comparativa entre a abordagem proposta nesta dissertação e o tipo de abordagem proposta na maioria das abordagens anteriores.

5.1 Descrição do Ambiente Experimental

O ambiente considerado para o desenvolvimento da abordagem é ilustrado na Figura 4. A configuração para o desenvolvimento da abordagem foi um cenário real onde clientes em redes sem fios acessaram o ambiente de grade dentro de uma rede estruturada. Este ambiente foi projetado para obter uma real avaliação de uma execução de *workflow* e casos relacionados.

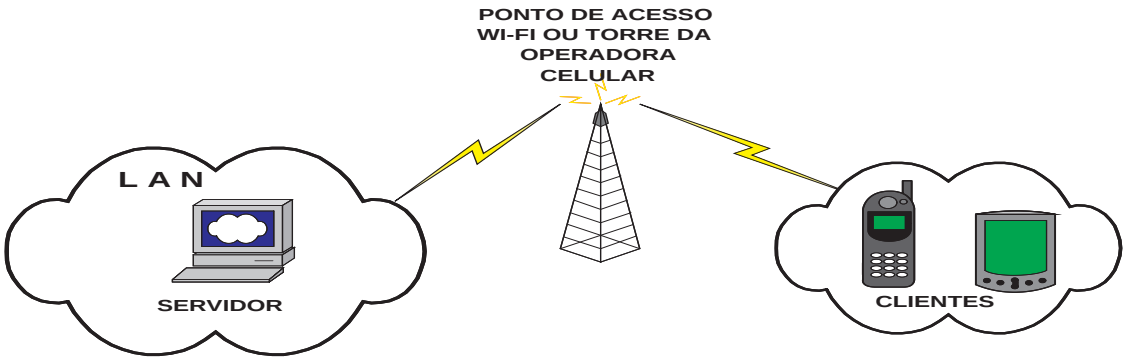

Figura 4: Cenário de Grade Móvel

Como visto anteriormente no capítulo 4, existe uma tendência na utilização de arquiteturas cliente-servidor na integração dos ambientes móveis com os ambientes de grade computacional [(CHU; HUMPHREY, 2004),(AHMAD et al., 2004), (GONZALEZ-CASTANO; VALES-ALONSO; LIVNY, 2003)]. Desta forma, nesta dissertação foi utilizado o modelo de comunicação cliente-servidor. Esta opção foi adotada considerando que, através da utilização deste modelo de comunicação, é possível usufruir do grande poder computacional oferecido pelas configurações de grade fornecendo serviços que exijam uma quantidade maior de processamento e recursos de diversos tipos. Desta forma, os dispositivos móveis passam a ser utilizados no modo de interação Interface (descrito anteriormente no capitulo 4) com os ambientes de alto desempenho fornecidos pela grade.

O servidor foi devidamente configurado e instalado com o Globus Toolkit versão 4.0.0 (GLOBUS, 2006). A escolha foi realizada pelo fato deste ser um dos pacotes mais completos, possuindo boa documentação, fornecendo a nova especificação WSRF (WSRF, 2004) para definição de serviços, voltado já para *Web Services* e ser disponibilizada de forma gratuita como uma versão estável pela Globus Alliance (ALLIANCE, 2006). Os serviços básicos foram criados

de um modelo de programação uniforme para que usuários possam ter seus requisitos atendidos, independentemente do tipo de sistema computacional empregado. Estes serviços oferecidos por este *middleware* possibilitam: segurança, locação e gerenciamento de recursos, gerenciamento de dados, comunicação entre sistemas heterogêneos, infra-estrutura de informações, detecção de falhas e portabilidade (GLOBUS, 2006).

A interação com o Globus Toolkit 4.0.0 e o servidor foi realizada através do pacote de software Java CoG kit (Java Commodity Grid Kit) (LASZEWSKI et al., 2001). Este pacote de software permite o desenvolvimento rápido de aplicações que interagem com todos os serviços Globus através de interfaces de programação desenvolvidas por este pacote. Além disso, Java CoG possui um grande conjunto de APIs com diversas funcionalidades.

Entre os trabalhos citados, (WASSON et al., 2004) mostra que os clientes móveis (isto é, PDAs) foram desenvolvidos utilizando o framework .NET da Microsoft (Microsoft Corporation 2006, 2006) e portanto eles são dependentes da plataforma do Windows CE utilizadas nos dispositivos PocketPCs. Por outro lado, nesta dissertação foram empregadas tecnologias que obtém mais portabilidade nos dispositivos móveis. Desta forma, os clientes móveis foram desenvolvidos usando J2ME (Java 2 Micro Edition) Wireless Toolkit versão 2.2 (Sun Microsystems, 2006b), por ser uma ferramenta que fornece facilidades como por exemplo, emuladores, documentação e portabilidade para um grande número de dispositivos móveis. As características desta ferramenta são descritas com mais detalhes no apêndice B.1.1.

Para prover a comunicação entre os clientes e o servidor foi utilizado o protocolo HTTP. O principal motivo que levou a adotar este protocolo de comunicação foi o fato deste protocolo ter suporte garantido em todos dispositivos móveis que possuem uma KVM configurada (TOPLEY, 2002). Por outro lado, as conexões usando socket ou datagramas não são suportadas em todas as MIDP dos dispositivos móveis existentes. Desta forma, os clientes J2ME comunicam-se com um servidor que emprega a tecnologia *servlet* usando o protocolo HTTP, assim como mostrado na Figura 5. A especificação *servlet* fornece uma API que permite tratar requisições HTTP, mais detalhes sobre *servlet* estão no apêndice B.1.5. Sendo assim, a abordagem proposta por

esta dissertação possuem portabilidade e flexibilidade. Em outras palavras, esta abordagem permite que diversos dispositivos móveis (isto é, celulares e PDAs) com diferentes plataformas comuniquem-se com o servidor. Desta forma, os dispositivos móveis ou clientes podem enviar pedidos de submissão de diferentes *workflows*, pedidos para saber informações de status de cada tarefa do *workflow* submetido e também requisições de outras informações utilizando o protocolo de comunicação HTTP.

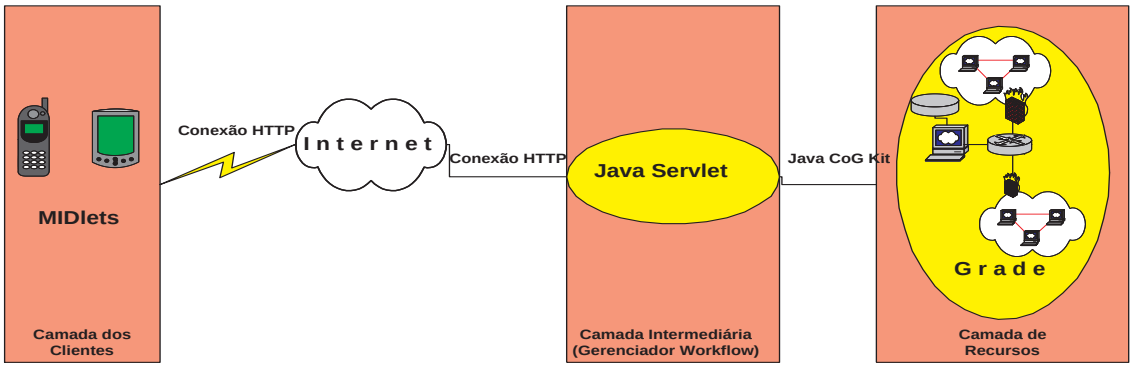

Figura 5: Cenário Servlet

A maior parte das abordagens mostradas na seção 4 não permite a submissão e monitoração de tarefas a partir de um telefone celular. (GONZALEZ-CASTANO; VALES-ALONSO; LIVNY, 2003) é única abordagem a permitir submeter e monitorar tarefas a partir dos dois tipos de dispositivos móveis, entretanto esta abordagem não emprega o conceito de *workflow*. A falta de abordagens que empregam telefones celulares acessando um ambiente grade é justificada pelo fato destes aparelhos serem muito mais limitados do que PDAs em quase todos aspectos. No entanto, a abordagem proposta nesta dissertação fornece uma interface com as mesmas funcionalidades para PDAs e telefones celulares. Desta forma, fornece ao usuário uma flexibilidade de escolher qual equipamento móvel utilizar, usufruindo das vantagens oferecidas por cada tipo de dispositivo em conformidade com as variadas necessidades de diferentes usuários. É importante salientar que algumas destas interfaces são adaptadas conforme as restrições de cada tipo de dispositivo. As diferenças ou adaptações das interfaces relacionadas aos diferentes tipos de aparelhos serão discutidas com mais detalhes na seção 5.3.

A utilização de telefones celulares permite atingir um número maior de pessoas, visto que

estes aparelhos são geralmente menores e mais leves que PDAs e portanto oferecendo uma maior facilidade em portá-los e além disso, eles possuem um custo financeiro menor de aquisição. Por outro lado, PDAs possuem menas limitações de recursos do que os telefones celulares e não existe a preocupação de custos financeiros relacionados a transferência de informações na rede sem fio. Vale a pena ressaltar que os telefones celulares mencionados na comparação anterior e empregados nesta dissertação são aqueles que utilizam a torre de transmissão de uma operadora de celular (neste caso, a operadora Claro) para transmitir dados na Internet utilizando a tecnologia de transmissão de dados GPRS (BETTSTETTER; VÖGEL; EBERSPÄCHER, 1999). Em outras palavras, a operadora possibilita o telefone celular estabelecer uma conexão com rede estruturada onde reside a configuração grade através de uma tarifação por volume de dados transmitidos.

No servidor, optou-se por arquivos XML para armazenar as informações de status de execução de cada tarefa do *workflow* de forma persistente, semi-estruturada e hierárquica. Desta forma, as informações correm menos risco de serem perdidas, fato que provavelmente ocorreria se o servidor armazenasse as informações somente na memória volátil (RAM) e porventura acontecesse alguma falha com o mesmo durante a extração das informações de status. Assim, quando o servidor recebe um pedido de informações de status, ele extrai as informações do arquivo XML, converte-as em texto e as retorna ao cliente.

A escolha do padrão XML (BRAY; PAOLI; SPERBERG-MCQUEEN, 1998) se deve também a outras vantagens oferecidas na sua utilização, por exemplo: uso de soluções tecnológicas não proprietárias; reuso das informações e apresentação das informações de modo flexível, uma vez que o conteúdo passa a ser independente da forma como ele é visualizado (por exemplo, o arquivo XML com informações de status pode ser convertido em um vetor de *strings*).

5.2 Arquitetura

A arquitetura da abordagem proposta é mostrada na Figura 6. Esta arquitetura possui dois componentes principais: o Portal, desenvolvido para os clientes móveis e o Gerenciador de

Workflow (isto é, o servidor), onde está instalado e configurado o Globus Toolkit. Nas subseções 5.2.1 e 5.2.2 são discutidos detalhes dos componentes da arquitetura, por exemplo: funcionalidade dos módulos, interação entre os módulos de um único componente, como também a interação entre os dois componentes e o *middleware* grade.

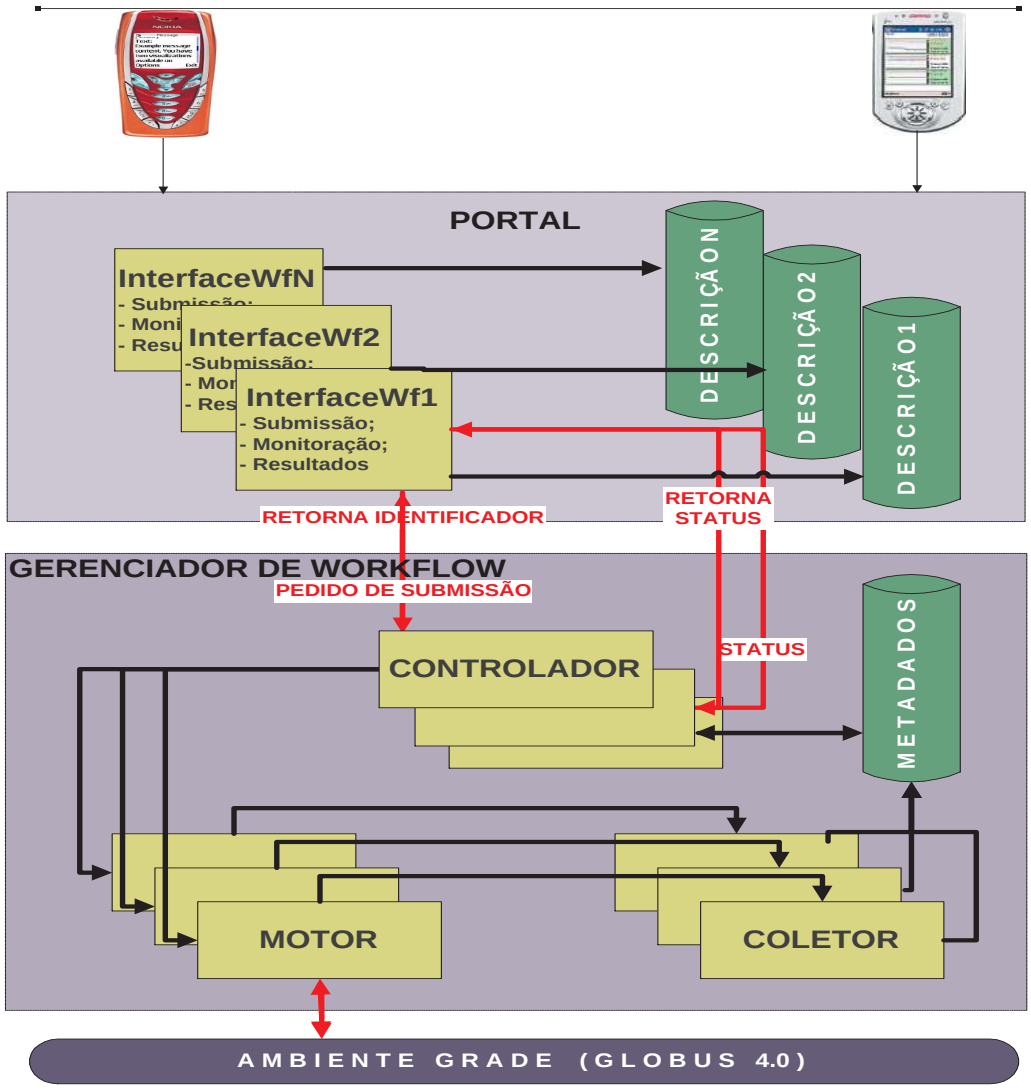

Figura 6: Arquitetura da Abordagem Proposta

5.2.1 Gerenciador de Workflow

O componente Gerenciador de Workflow foi projetado e implementado para processar os pedidos (por exemplo, submissão, monitoração e visualização de resultados de *workflows* submetidos) vindos dos dispositivos móveis, gerenciar e controlar a execução de tarefas e coletar

informações relacionadas à execução destas tarefas, realizando todas estas funcionalidades de forma transparente para o cliente móvel.

Este componente se encontra no servidor e possui três módulos que são: Controlador, Motor e Coletor. O Gerenciador de Workflow é responsável por interações com o componente Portal e também com o ambiente de grade através do *middleware* Globus. Exemplos de interações são: pedido de submissão recebido de um *workflow* qualquer, pedido de status do *workflow* submetido e também interações com os serviços do Globus (por exemplo, MDS e GRAM) (FOSTER, 2005).

Para cada módulo deste componente foi criada uma classe usando o J2SE (Sun Microsystems, 2006c), no caso do Controlador, e a API oferecida pelo Java CoG Kit, chamada Karajan Workflow Engine (LASZEWSKI; HATEGAN, 2005), usada nos módulos Motor e Coletor, como pode ser visto no diagrama de classes no apêndice C. Karajan é um motor de execução e linguagem *workflow* versátil e fácil de usar, que foi descrita na seção 3.5 do capítulo 3. Optou-se pelo seu uso devido ao fato de apresentar todas as estruturas de definição de *workflow* (por exemplo, seqüencial, paralelo, iteração ou *loop* e roteamento) contidas também em outras linguagens, estruturas de definição avançadas (por exemplo, lista, *range* e índice *hash*) para tarefas repetitivas que não estão nas linguagens *workflows* disponíveis no meio científico e também suporta variados tipos de aplicação grade, como mostrados na Tabela 1 e Tabela 2 do capítulo 3. Em adição, esta API oferece tratamento de falhas, checkpointing, execução distribuída em nodos grade distintos e também permite extensão de serviços Globus secundários através da camada núcleo de abstração do Java CoG Kit.

Desta forma, é possível definir *workflows* e as relações mais complexas de suas tarefas para executar em ambientes de grade. Através da utilização da linguagem Karajan é possível definir *workflows* complexos e de variados tipos utilizando a variedade de estruturas que são suportadas nesta linguagem e motor de execução de *workflow*. Karajan apresenta outras vantagens em relação às demais linguagens e motores de execução *workflow*, como por exemplo: único motor de execução dentre os demais que fornece uma API como uma forma de abstração (vide Tabela 1

no capitulo 3) para o desenvolvimento de aplicações grade e também o único motor de execução que atualmente permite integração com diferentes *middlewares*, entre os quais estão: diversas versões do Globus Toolkit (por exemplo, 4.0 e 3.0) e Condor. Em outras palavras, utilizando esta API e linguagem foi possível desenvolver classes e métodos que submetem *workflows* ao *middleware* Globus, invocando seus principais serviços (como por exemplo, GRAM e MDS).

Nos tópicos seguintes são descritos os módulos que estão disponíveis neste componente. Conforme mostrado na Figura 6, os módulos são:

- **Controlador**: recebe o pedido de submissão vindo do Portal, retorna o identificador do *workflow* que está sendo executado para a interface contida no Portal. O Identificador é um atributo numérico que identifica de forma única a submissão recebida por este componente em um determinado instante. Desta forma, este atributo permite controlar e ordenar os pedidos de submissão de *workflow* vindos de algum dispositivo móvel. Sendo assim, este atributo é passado até o módulo Coletor, onde o mesmo cria o arquivo XML de *checkpoint* para armazenar as informações de status, utilizando como o nome deste arquivo de *checkpoint* o valor contido no próprio atributo Identificador (isto é, se o valor do Identificador for 1043, o nome do arquivo de *checkpoint* será 1043.xml). Assim, quando o Controlador recebe um pedido de status, o Identificador vem junto com este pedido. Através do Identificador, o Controlador pode localizar o arquivo XML com as informações de status e extraí-las, enviando-as de volta para a Interface do Portal que fez o pedido. O Identificador é gerado de forma incremental pelo Controlador (isto é, a cada pedido recebido é incrementado 1 unidade a este atributo). Este módulo também cria uma instância da classe Motor, passando a localização do arquivo XML com a definição do *workflow* submetido e também o Identificador. Este módulo é implementado em *multithreading*, assim cada pedido de submissão é processado por uma Thread diferente;

- **Motor**: interpreta os arquivos de definição de *workflow* em formato XML com descrição de um determinado *workflow* para saber como deverá submeter e controlar a execução de todas as atividades ou tarefas (por exemplo, controle de fluxo e dados e invocação de

ferramentas computacionais) que são partes do *workflow* executado no ambiente de grade. Os arquivos XMLs de descrição do *workflow* estão especificados usando a linguagem Karajan. Um exemplo de um arquivo desta natureza pode ser visto no apêndice D. Este módulo cria uma instância da classe Coletor, passando como atributo o Identificador do *workflow* submetido. A submissão é dependente da disponibilidade do nodo grade e das restrições de execução (por exemplo, dados não podem ser movidos de um lugar para outro);

- **Coletor**: este módulo é responsável por processar os eventos gerados pela execução da *workflow*. Em outras palavras, ele analisa as informações de status de execução *workflow* e armazená-las em um arquivo XML em um repositório de metadados. A estrutura do arquivo XML com as informações do status de execução do *workflow* pode ser vista na Figura 7.

```
<project number= "0" workflow_name= "Complete-
Genome">
    <UID8 time="15232">COMPLETED</UID8>
    <UID9 time="9904">COMPLETED</UID9>
    <UID10 time="8830">COMPLETED</UID10>
    <UID12 time="16176">COMPLETED</UID12>
    <UID13 time="9585">COMPLETED</UID13>
    <UID15 time="">RUNNING</UID15>
    <UID16 time="9482">COMPLETED</UID16>
    <UID17 time="">FAILED</UID17>
    <UID18 time="">FAILED</UID18>
</project>
```

Figura 7: Arquivo XML de Checkpoint Workflow

O atributo **number** da tag **project** armazena o valor do atributo Identificador. A tag **UIDX** significa a identificação de cada tarefa *workflow*, visto que neste exemplo existem 9 tarefas. O atributo **time** de cada tag UID significa o tempo de execução daquela tarefa em milisegundos.

O diagrama de seqüência mostrado na Figura 8 descreve o comportamento do componente

Gerenciador Workflow quando o mesmo recebe um pedido de submissão de um exemplo *workflow* disponibilizado para execução no Portal. Os passos seguidos pelo componente para a correta submissão do *workflow* e a geração do arquivo XML com as informações de status são:

- No passo 1, um pedido de submissão é recebido e a classe Controlador inicia uma thread que exclusivamente processa este pedido de submissão *workflow*.
- O método **load** carrega o arquivo de definição do *workflow* na linguagem Karajan e extrai as informações de como deve ocorrer o fluxo de execução do *workflow* (passo 2). Logo em seguida, o metódo **start** (passo 3) inicia a execução do pedido de submissão do *workflow* através da classe Motor. A localização do arquivo de definição do *workflow* em linguagem Karajan é passado como atributo neste método;
- Logo após o começo da execução no passo 4, o método **statusMonitoringEvent** da classe **ColetorComGen** é executado. Neste momento, os eventos da execução do *workflow* são gerados pela instância **Motor** e analisados pela instância **ColetorComGen**. Desta forma, as informações de status são coletadas;
- No passo 5, a instância **ColetorComGen** cria uma instância da classe **DOM** para geração ou atualização do arquivo XML com as informações de status processadas pelo método **statusMonitoringEvent**. O método **updateXMLfile** é responsável pela atualização do status no arquivo XML.

O Gerenciador de Workflow fornece a transparência necessária ao usuário móvel para executar *workflows* em uma configuração de grade utilizando os serviços do *middleware* grade, sem conhecimento de como e onde são empregadas as políticas de gerenciamento de recursos, segurança, acesso aos dados e balanceamento de carga. Assim, usuários móveis podem acessar serviços do Globus transparentemente. Estes serviços fornecem um modo confiável, seguro, eficiente, transparente e robusto de acesso aos recursos da configuração grade para execução de *workflows* através de dispositivos móveis. Esta transparência não é fornecida em (GONZALEZ-CASTANO; VALES-ALONSO; LIVNY, 2003), onde os usuários dos dispositivos móveis precisam

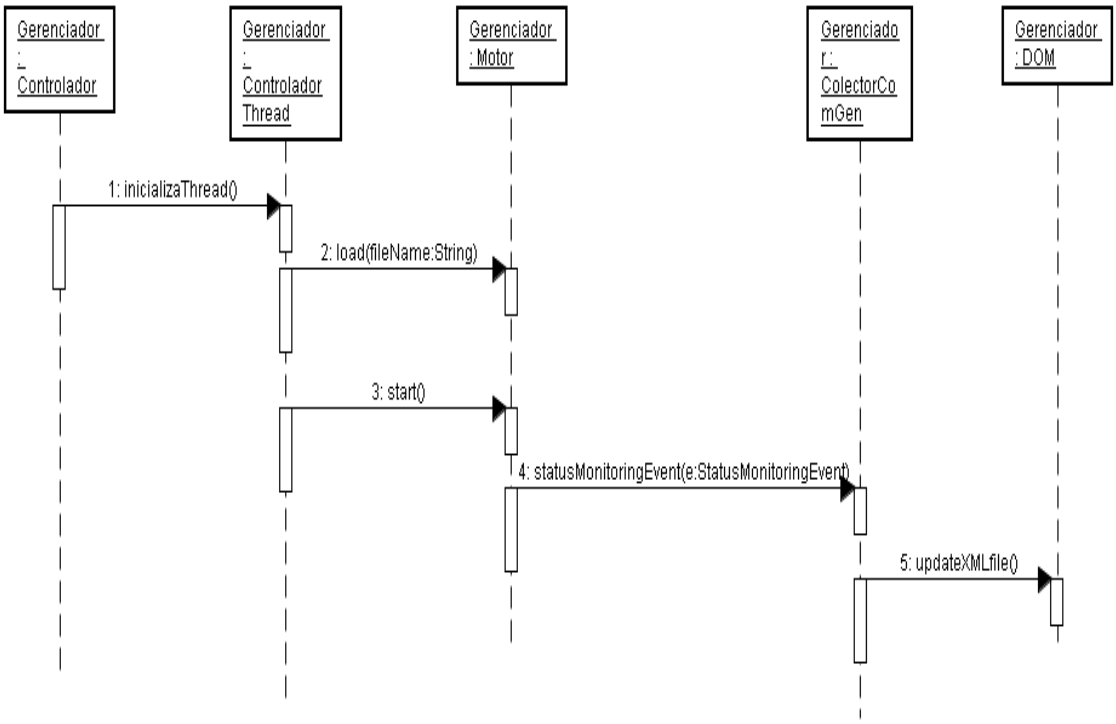

Figura 8: Diagrama de Seqüência de um pedido de submissão recebido

conhecer a sintaxe de comandos e políticas do gerenciador de tarefas Condor para submeter e monitorar as tarefas no ambiente de grade.

5.2.2 Portal

Além da submissão de *workflow* e da visualização dos resultados a partir do dispositivo móvel. O componente Portal foi projetado para fornecer interfaces com novas funções e mais informações detalhadas do que as interfaces apresentada nos trabalhos anteriores (GONZALEZ-CASTANO; VALES-ALONSO; LIVNY, 2003; CHU; HUMPHREY, 2004; BROOKE; PARKIN, 2005; HWANG; ARAVAMUDHAM, 2004; AHMAD et al., 2004). As novas funções e informações fornecidas pelas interfaces disponibilizadas no Portal são:

- Acessar a descrição de *workflows* e suas tarefas. Em outras palavras, fornecendo acesso à descrição das ferramentas computacionais (por exemplo, banco de dados e software) que são invocadas pelas tarefas e a função de cada tarefa *workflow*. Esta funcionalidade do Portal ajuda usuários selecionarem quais *workflows* podem resolver seus problemas. A

descrição de cada *workflow* do Portal está estruturada no padrão XML;

- Monitorar passo a passo a execução de cada *workflow* submetido através do fornecimento de informações de status de cada tarefa do *workflow* (por exemplo, executando, falhou ou terminou);

- Visualizar o tempo que cada tarefa *workflow* levou para executar e também o tempo total de execução do *workflow*;

- Visualizar possíveis problemas que venham a ocorrer durante a execução das tarefas do *workflow* (por exemplo, formato inadequado dos arquivos de entrada);

- Visualizar execuções de *workflows* de pedidos anteriores.

A submissão de um determinado *workflow* pode ser mais bem visualizada no diagrama de seqüência UML da Figura 9. Os passos da submissão são descritos logo abaixo:

- No passo 1, inicia a interface de um determinado *workflow*, desenhando-o na tela do dispositivo e disponibilizando as opções do Menu com as funcionalidades do Portal apresentadas anteriormente;

- No passo 2, no instante que o cliente submete o *workflow*, o método **submitWorkflow** envia o pedido de submissão ao módulo **Controlador**;

- No passo 3, uma instância **ControladorThread** é criada para submeter *workflow* à configuração grade;

- No passo 4, depois do *workflow* submetido à configuração grade (estes passos estão descritos no diagrama de seqüência da Figura 8), a instância **ControladorThread** envia o Identificador para o cliente móvel que enviou o pedido de submissão.

A monitoração do *workflow* submetido é iniciada sem qualquer interação ou intervenção do usuário com a interface do Portal, ou seja, de forma automática. Desta forma, a monitoração

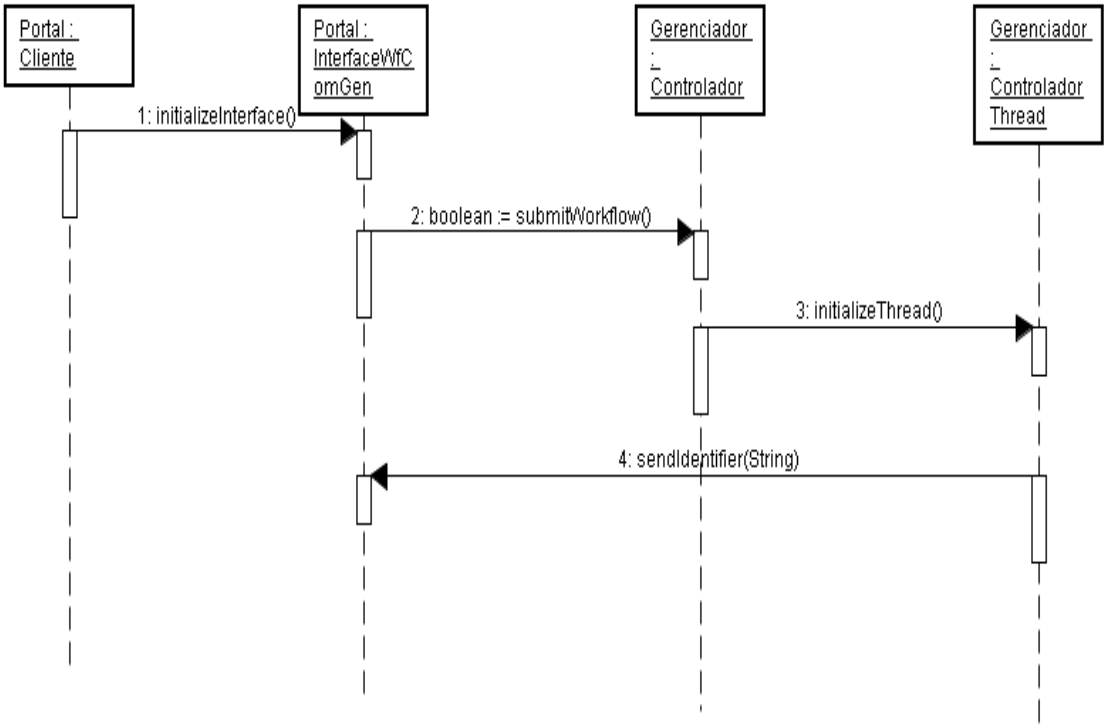

Figura 9: Envio do pedido de submissão Workflow

começa no instante em que a interface do Portal recebe de volta o Identificador da submissão do *workflow* e vai até o momento em que todas as tarefas *workflows* terminarem a execução. A monitoração da execução do *workflow* foi realizada enviando pedidos de status em um intervalo de tempo pré-definido inicialmente de 8 segundos. Em outras palavras, os pedidos de status dos *workflows* submetidos foram enviados a cada 8 segundos. Desta forma, a monitoração ocorre de forma constante na interface do dispositivo. O recebimento do Identificador do *workflow* submetido confirma que o pedido de submissão foi recebido e iniciado com sucesso. O diagrama de seqüência UML da Figura 10 mostra os passos no processo de monitoração *workflow*. Os passos da monitoração são descritos logo abaixo:

- No passo 1, logo depois de receber o Identificador, o método **monitorWorkflow** é executado passando como atributo o Identificador da submissão **workflow** que se deseja monitorar, em seguida é enviado o pedido de status com o Identificador do **workflow** submetido ao módulo **Controlador**;

- No passo 2, uma instância de **ControladorThread** é criada para processar o pedido de

status do *workflow*;

- No passo 3, o método **recovery** identifica através do Identificador o arquivo XML de *checkpoint* para extrair as informações de status do *workflow*;
- No passo 4, depois de extrair as informações, o método **sendStatus** envia as informações de status de cada tarefa *workflow* de volta ao cliente que fez o pedido de status.

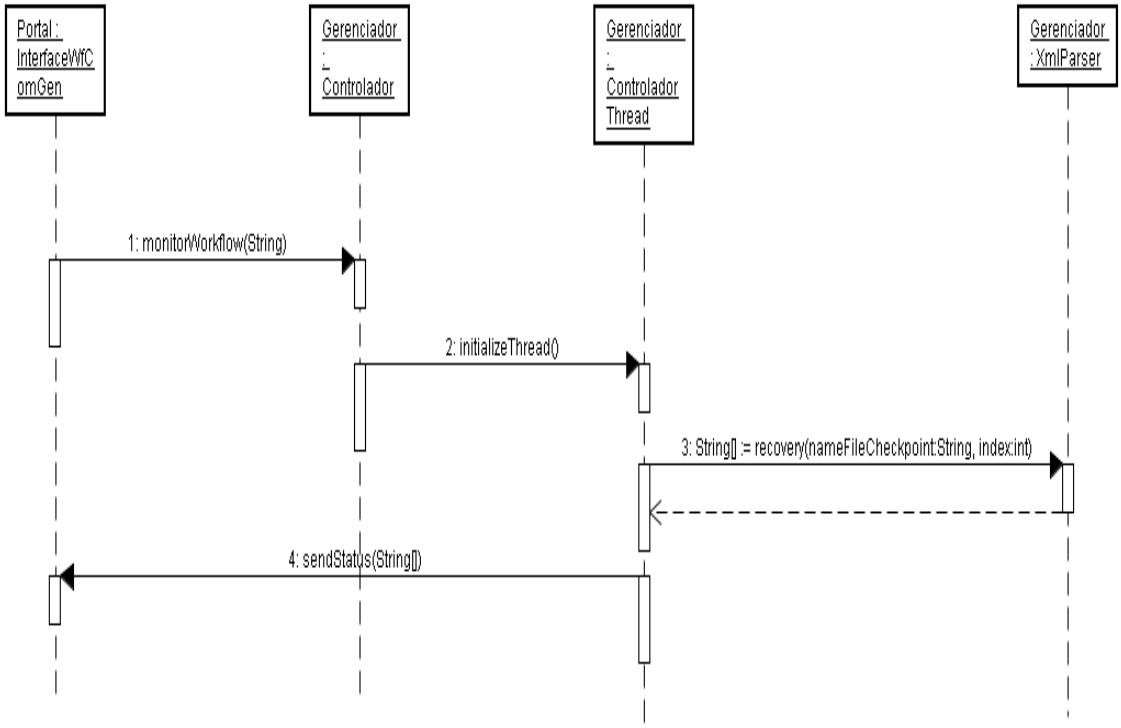

Figura 10: Envio do pedido de monitoramento Workflow

O fornecimento de informações relevantes (por exemplo, status das tarefas de *workflows*) torna-se interessante quando realizada de uma forma mais flexível, eficiente e obedecendo a necessidade de cada usuário em determinado instante de tempo, conforme destacado em (WANG; WANG, 2005). Desta forma, a abordagem proposta nesta dissertação permite que o usuário especifique o intervalo de monitoração antes da submissão de um determinado *workflow*. Esta flexibilidade permite que o usuário do dispositivo móvel ajuste a monitoração do *workflow* de acordo com a atual capacidade do aparelho. Por exemplo, o usuário pode colocar um intervalo maior de monitoração do que ele costuma adotar, devido ao fato do dispositivo móvel apresentar pouca energia na bateria, uma vez que quanto maior for o acesso na rede sem fio, maior será

o gasto de energia da bateria. Já no caso específico de telefones celulares, outra razão para adaptar um intervalo de monitoração maior seria a pouquíssima quantidade de crédito para fazer a monitoração, tornando interessante que o cliente móvel efetue menos pedidos de status ao servidor.

Existe uma outra forma para transferir as informações de status de execução do *workflow* submetido até os dispositivos móveis. A outra opção seria transferir o arquivo XML para o cliente móvel via HTTP, e logo que o cliente recebesse o arquivo extrairia as informações do XML e as apresentaria na interface para visualização do usuário. No entanto, esta solução acrescenta uma carga de processamento maior ao Portal, degradando seu desempenho e conseqüentemente ocasionando um maior consumo de energia da bateria destes dispositivos. Visto que o uso de CPU e acesso ao meio de comunicação sem fio são fatores significativos no consumo de energia da bateria destes dispositivos móveis, assim como mostrado por (RONG; PEDRAM, 2003), (KRAVETS; SCHWAN; CALVERT, 1999) e (MOHAPATRA et al., 2005). Portanto, a escolha realizada para deixar que o componente Gerenciador Workflow extraísse as informações do arquivo XML e as enviasse ao dispositivo, fornece ao usuário destes dispositivos uma interface que degrade menos o desempenho destes aparelhos e, consequentemente proporcione maior economia no consumo da energia.

As abordagens apresentadas em (CHU; HUMPHREY, 2004; HWANG; ARAVAMUDHAM, 2004) não apresentam a funcionalidade de monitoração de tarefas. Já as abordagens de (BROOKE; PARKIN, 2005; AHMAD et al., 2004) permitem a monitoração de tarefas, mas não a monitoração de várias tarefas em uma única interface organizada e adaptada às limitações de tela desses dispositivos; somente uma tarefa por vez na interface do dispositivo móvel é permitido nestas abordagens. Uma monitoração *workflow* em uma única interface reduz a interação do usuário no dispositivo móvel para acompanhar a execução do *workflow*, visto que estes geralmente são compostos por várias tarefas do (isto é, mais que 2 tarefas) e permite acompanhar a execução das várias tarefas *workflows* simultaneamente. Desta forma, a interface desenvolvida nesta dissertação pretende fornecer uma interface mais automatizada, organizada e transparente na forma de apresentar as informações de status de cada tarefa e assim proporcione uma interface

mais amigável que os trabalhos mostrados por (BROOKE; PARKIN, 2005; AHMAD et al., 2004), onde é necessário que o usuário faça o pedido de status de cada tarefa em momentos distintos através da interface.

As limitações observadas nos dispositivos de entrada e saída destes aparelhos [(LITKE; SKOUTAS; VARVARIGOU, 2004),(FORMAN; ZAHORJAN, 1994),(PARK; KO; KIM, 2003), (BRUNEO et al., 2003)] torna muito difícil a composição de *workflows* a partir desses dispositivos. Esta composição requer uma tela com melhor resolução de vídeo, assim como observado por (HOHEISEL, 2005), onde é disponibilizada um interface sofisticada para desenhar, projetar e validar *workflow*. Desta forma, nesta dissertação não se trata a composição de *workflows* a partir destes dispositivos. Sendo assim, os *workflows* disponibilizados no Portal estão prontamente definidos ou compostos como scripts em formato XML da linguagem Karajan.

5.3 Protótipo Implementado

A interação dos dispositivos móveis com a grade computacional possibilita a alguns usuários de certas áreas um ganho de produtividade fora dos locais habituais de trabalho, fornecendo um ambiente com alta capacidade de processamento e uma grande quantidade de recursos e serviços que permite resolver problemas complexos. Por exemplo, um biólogo pode sequenciar um DNA de alguma determinada espécie; um físico pode analisar os dados gerados de uma configuração de grade; um químico pode calcular o risco ambiental causado pelo transporte de um poluente na atmosfera; e mais alguns tipos de aplicações podem ser fornecidos a outros profissionais.

O protótipo desenvolvido nesta dissertação será tratado com dois exemplos de *workflows* de bioinformática para validar a abordagem proposta. O motivo que levou a escolher bioinformática é pelo fato dos pesquisadores desta área necessitarem de ferramentas ou sistemas mais automatizados para facilitar a analise de seqüências DNA, RNA e proteínas, assim como mostrado por (LEMOS, 2004). Importante ressaltar que durante a fase de análise, os pesquisadores utilizam diversos programas de computador e um grande volume de dados que em alguns ca-

sos, estão armazenados em locais geograficamente distintos. Como pode ser visto em [(OINN et al., 2004),(STEVENS; ROBINSON; GOBLE, 2003), (CANNATARO et al., 2004), (LEMOS, 2004)], *workflows* têm sido uma solução bastante adotada para auxiliar os biólogos a lidar com a complexidade e o volume crescente destes dados.

O primeiro exemplo de *workflow* (vide Figura 11) tem como objetivo pesquisar nucleotídeos em três diferentes bancos de dados de proteínas armazenados no servidor. As descrições dos recursos utilizados neste *workflow* estão detalhadas na Figura 46 do apêndice E. Já o segundo exemplo de *workflow* (vide Figura 12) foi caracterizado pela análise da bactéria *Gluconacetobacter diazotrophicus* do projeto Genoma (LEMOS, 2004). A descrição de todos programas e banco de dados utilizadas em cada tarefa deste *workflow* estão descritos na Figura 47 do apêndice E. As subseções seguintes descrevem as interfaces dos dois diferentes tipos de dispositivos móveis empregados nesta dissertação.

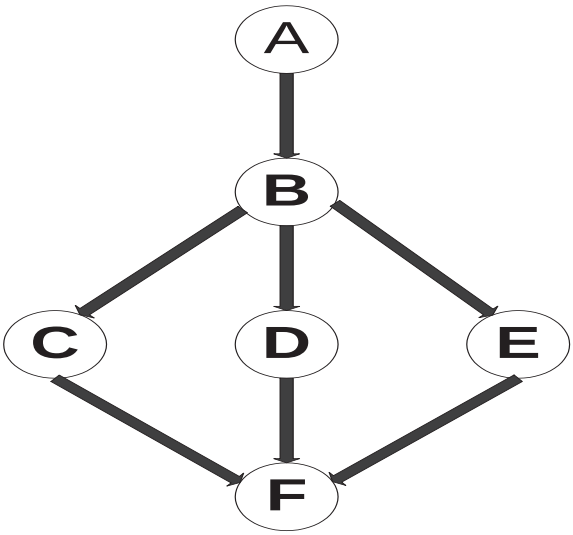

Figura 11: Primeiro Exemplo de Workflow

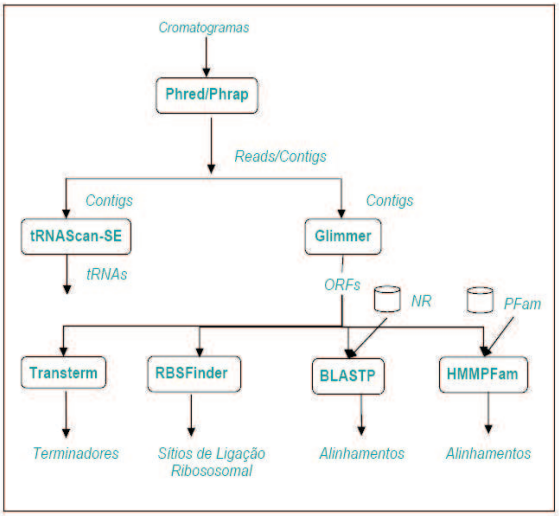

Figura 12: Segundo Exemplo de Workflow (LEMOS, 2004)

A tabela 4 contém as características de hardware e pacotes de software do servidor e dos clientes utilizados no protótipo implementado:

Tabela 4: Características de hardware e software do ambiente experimental

Nodos	Modelo	Processador	Memória	SO	JVM	Rede
PDA	Palm Tungsten C	400 MHz	64 MB	Palm OS 5.2.1	WEME J9 (IBM)	11 Mbps
Celular	Motorola C385	Não conhecido	3MB	Próprio da Motorola	Próprio da Motorola	32 - 48 Kbps
Servidor	AMD Duron	1200 MHz	512 MB	Linux Red Hat 9.0	J2SE 1.5.0_04-b05	100 Mbps

5.3.1 Interfaces nos PDAs

A figura 13 mostra a tela inicial do Portal. Através dessa tela o usuário pode escolher qual *workflow* vai submeter ao ambiente grade. A figura 14 apresenta as opções contidas no menu do Portal. Dentre elas estão: *Running*, *Show Results*, *Show Problems*, *Show Execution Time*, *Show Description* e *Show Previous Execution*. As interfaces dos diferentes exemplos de *workflow* possuem um menu com as mesmas opções, fornecendo um Portal com interfaces padronizadas para diferentes *workflows*.

Para submeter o *workflow*, o usuário deve selecionar a opção *Running*. Depois de selecionar esta opção, o usuário deve especificar os dados de entrada para a execução do *workflow*. No primeiro exemplo, o usuário vai especificar o diretório do arquivo onde se encontra o texto a ser pesquisado nos bancos de dados de nucleotídeos. Uma vez que digitação de textos utilizando o teclado pequeno destes equipamentos torna bastante difícil e árdua a entrada destes dados. Também no segundo exemplo, o usuário vai especificar o diretório onde se encontram os vários arquivos em formato de cromatogramas nos quais será feita a análise e anotação da bactéria *Gluconacetobacter diazotrophicus*, como pode ser visto na Figura 15.

Logo após a especificação dos parâmetros de entrada, o *workflow* começa a executar e a interface de monitoração é ativada para acompanhar o progresso da execução do *workflow*, como pode ser visto nas Figuras 16 e 17. As informações de status de cada tarefa são representa-

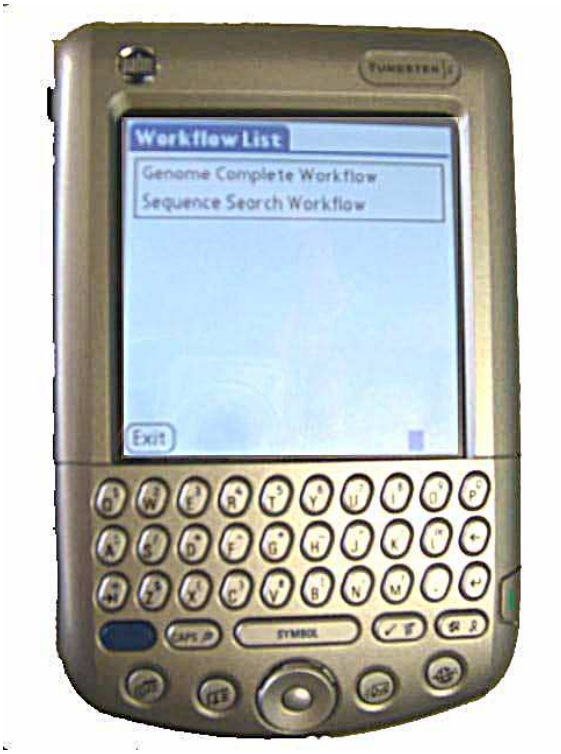

Figura 13: Tela Inicial

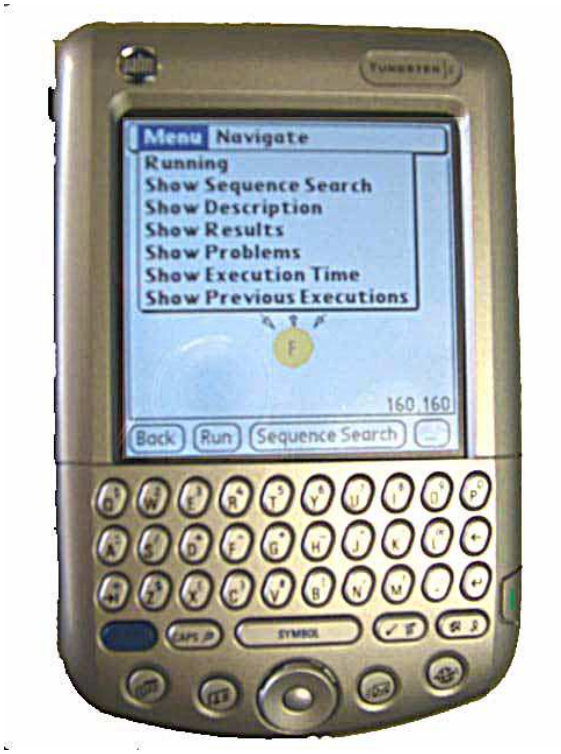

Figura 14: Menu

das através de um padrão de cores preenchidas nos círculos da estrutura Não-DAG da interface do *workflow*, assim como descrito na seção anterior. Desta forma, o usuário pode monitorar simultaneamente as diversas tarefas do *workflow* por uma interface única e padronizada, possibilitando uma monitoração com mais vivacidade ou animação. Importante lembrar que informações são atualizadas em um intervalo de tempo pré-definido inicialmente de 8 segundos durante todo processo de monitoração, sendo que o usuário pode especificar outro intervalo de acordo com as condições atuais do aparelho. Quando a execução do *workflow* termina, uma mensagem avisando o término de execução aparece na tela do dispositivo. Além disso, uma outra forma de notificar ou alertar o usuário do término da execução é através de um alerta utilizando som por certo período de tempo. Desta forma, o usuário pode submeter o *workflow* e não precisa ficar observando na interface do aparelho a todo momento para averiguar se a execução terminou, basta que ele deixa aparelho em um local próximo que permita perceber o alerta.

A funcionalidade de monitoramento *workflow* possui uma interface com as seguintes características no PDA:

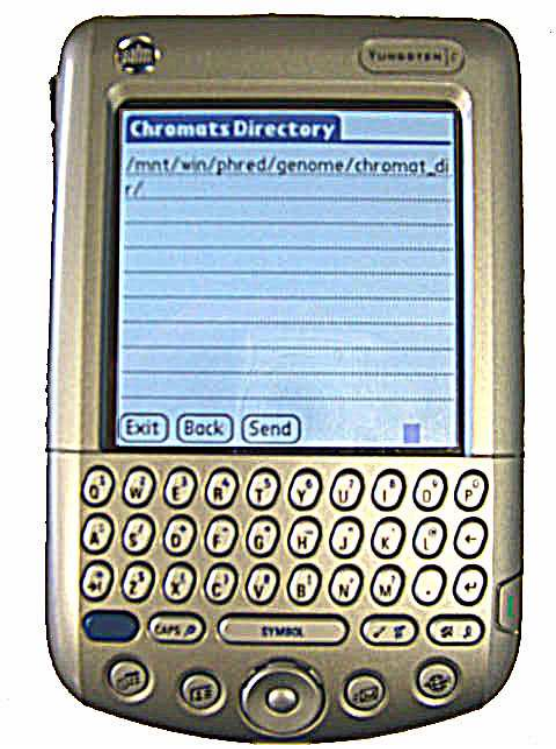

Figura 15: Especificar Localização dos Dados de Entrada

- **Estrutura única**: foi implementada uma interface que representasse o *workflow* graficamente em uma estrutura única através do formalismo Não-DAG. Uma API desenvolvida em J2ME (Robert Esser, 2006) foi utilizada, modificada ou adaptada para projetar e desenhar a estrutura do formalismo Não-DAG nos dispositivos móveis. Um estrutura única de representação do status das tarefas *workflows* fornece uma forma organizada das informações desta interface;

- **Adaptada ou Otimizada**: Dentre as opções mostradas na seção 3.3 desta dissertação para representação gráfica de *workflow*, o formalismo Não-DAG se mostrou o mais ideal para abordagem proposta nesta dissertação, por ser flexível no sentido de permitir todas as estruturas (por exemplo, seqüencial, loops e paralelismo) também contidas em outras opções de formalismo e por apresentar uma representação gráfica simples, ou seja, com o mínimo de elementos gráficos onde círculos ou nodos indicam tarefas e arcos indicam transição ou dependências entre essas tarefas. Desta forma, a representação gráfica fornecida pelo formalismo Não-DAG é interessante em dispositivos (isto é, PDAs e celulares) que possuem tamanho de tela reduzida;

- **Padronizada**: interface de monitoramento tem uma padrão de cores em cada círculo que indica o status atualizado de cada tarefa. Ou seja, quando a interface *workflow* do Portal recebe as informações de status de cada tarefa do *workflow* submetido, ela traduz estas informações em cores que são então preenchidas nos correspondentes círculos que representam as tarefas. Por exemplo, quando algum círculo está preenchido com a cor amarela significa que aquela tarefa não foi ainda submetida, a cor cinza significa que a tarefa está executando atualmente, cor azul significa que a tarefa terminou sua execução e a cor vermelha significa que a tarefa falhou ao executar por conseqüência de falha de hardware ou software. Assim, os usuários móveis estão aptos a monitorar uma execução *workflow* com mais animação ou vivacidade.

Figura 16: Monitoramento do Exemplo 1

Figura 17: Monitoramento do Exemplo 2

Workflows considerados de grande porte, ou seja, que possuem uma grande quantidade de tarefas, dificultam a implementação de uma interface de monitoramento para estes *workflows* em um dispositivo móvel que possui um tamanho de tela reduzida. A solução para este tipo de *workflow* foi representar um subconjunto de tarefas como uma única tarefa na interface de monitoramento, chamada tarefa simbólica. Os critérios adotados para selecionar as tarefas

que fazem parte deste subconjunto são: as tarefas devem ter dependências entre si e serem tarefas com menos relevância naquele *workflow* específico. Se o usuário desejar acompanhar a execução daquele subconjunto de tarefas representadas por uma única tarefa, deve pressionar com o lápis ótico do dispositivo sobre a tarefa simbólica e logo em seguida aparecerá uma tela com um *sub-workflow*, representando o subconjunto destas tarefas.

Assim, o tamanho que o *workflow* ocupa na tela do dispositivo pode ser simplificado sem que a funcionalidade de monitoramento das tarefas seja afetada ou reduzida. Um exemplo deste caso acontece no segundo exemplo *workflow* do Portal, onde a tarefa A1 representa o fluxo de execução seqüencial de três tarefas que invocam três softwares diferentes (isto é, phrep, phd2fasta, phrap (GREEN..., 2006)). A Figura 19 representa o *sub-workflow* do *workflow* apresentado na Figura 18. Uma outra solução seria representar todo o *workflow* em uma resolução bem menor na interface do dispositivo, mas isto dificultaria muito para o usuário identificar que tarefa cada circulo representa.

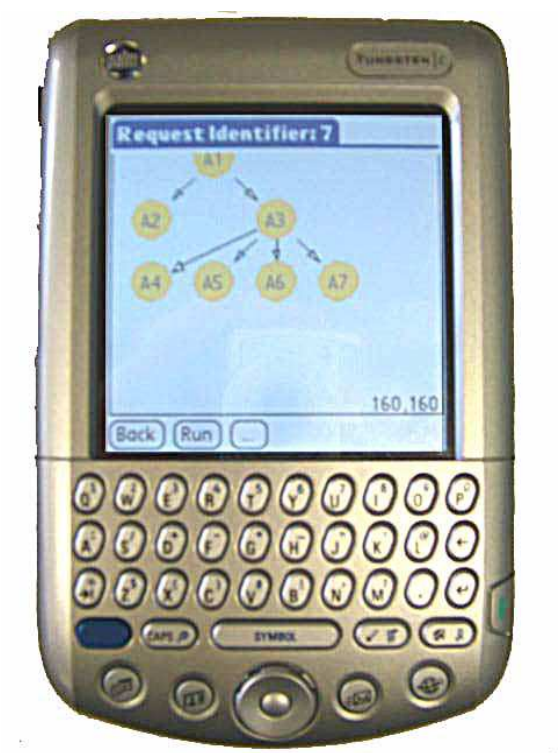

Figura 18: Workflow Completo

Figura 19: Sub-Workflow

A interface do Portal permite que o usuário possa visualizar os resultados parciais e totais obtidos em todas as tarefas dos exemplos de *workflows* contidos no Portal, através da opção

Show Results do menu. A Figura 20, mostra como o usuário pode selecionar a tarefa cujo o resultado deseja visualizar através de uma lista. Na Figura 21 ilustra a visualização do resultado da tarefa selecionada na tela do dispositivo. Os exemplos de *workflows* implementados nesta dissertação produzem resultados em arquivos com formato texto.

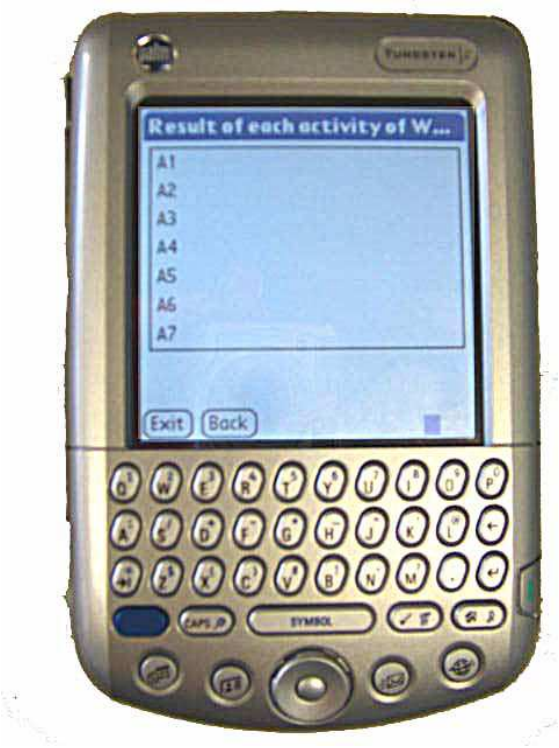

Figura 20: Listas das Tarefas Workflow

Figura 21: Resultado de uma tarefa

O tamanho reduzido e a baixa resolução da tela desses dispositivos, principalmente celulares, dificulta a visualização de resultados produzidos por alguns *workflows* de grade a partir destes aparelhos. Desta forma, é necessária a otimização de resultados produzidos pela execução de *workflows* ou/e adaptação das ferramentas de visualização para o contexto destes pequenos aparelhos (por exemplo, como realizado em (AHMAD et al., 2004)). Geralmente, *workflows* produzem resultados da seguinte forma: um arquivo final ou vários arquivos durante sua execução. Os arquivos de resultado geralmente são do tipo texto ou gráfico (por exemplo, exportado como uma figura no formato JPEG, GIF ou PNG).

Com objetivo de alcançar uma maior otimização dos resultados produzidos pelo *workflows* implementados nesta dissertação e desta forma, fornecendo uma melhor visualização dos mesmos para os usuários dos equipamentos portáteis, foi realizada uma análise minuciosa dos ar-

quivos gerados em várias execuções diferentes desses exemplos de *workflows*. Desta forma, foi observado que alguns arquivos mostravam textos que se repetiam e não apresentavam informações relevantes para análise daquela execução de *workflow*. Um exemplo deste tipo de informação foi encontrado nos cabeçalhos destes arquivos, onde eram descritos versão, ano de desenvolvimento e nome dos responsáveis pela ferramenta, conforme exemplificado na Figura 22. Desta forma, estas informações foram retiradas dos resultados e permaneceram somente as informações que pudessem possibilitar a análise deste profissional sem qualquer informação desnecessária, conforme mostrado na figura 23.

Em outras palavras, foi implementado um filtro onde os arquivos foram alterados, retirando-se estas informações sem importância. Desta forma, esta filtragem permite uma interface com textos reduzidos. Consequentemente, transferindo informações estritamente necessárias para o compreendimento dos resultados produzidos pelo *workflow* submetido. Em contrapartida, a redução dos arquivos traz como vantagem um menor tempo de transferência de arquivos para os dispositivos móveis, visto que depois da filtragem os arquivos ficam com menor tamanho.

Vale a pena ressaltar que o método empregado nesta dissertação para otimizar os resultados é específico e adequado para os exemplos de *workflows* aqui implementados. Portanto, este método talvez não se aplique em outros resultados produzidos por diferentes *workflows* de grade computacional. Desta forma, é necessário um estudo detalhado de cada conjunto de resultados para assim definir a maneira ou metodologia de otimização a ser empregada.

```
tRNAscan-SE v.1.23 (April 2002) - scan sequences for transfer RNAs

  Please cite:
        Lowe, T.M. & Eddy, S.R. (1997) "tRNAscan-SE: A program for
        improved detection of transfer RNA genes in genomic sequence"
        Nucl. Acids Res. 25: 955-964.

  This program uses a modified, optimized version of tRNAscan v1.3
  (Fichant & Burks, J. Mol. Biol. 1991, 220: 659-671),
  a new implementation of a multistep weight matrix algorithm
  for identification of eukaryotic tRNA promoter regions
  (Pavesi et al., Nucl. Acids Res. 1994, 22: 1247-1256),
  as well as the RNA covariance analysis package Cove v.2.4.2
  (Eddy & Durbin, Nucl. Acids Res. 1994, 22: 2079-2088).

-------------------------------------------------------------
Sequence file(s) to search:  /mnt/win/phred/genome/workflow/seqs_fasta.contigs
Search Mode:                 Eukaryotic
Results written to:          Standard output
Output format:               Tabular
Searching with:              tRNAscan + EufindtRNA -> Cove
Covariance model:            TRNA2-euk.cm
tRNAscan parameters:         Strict
EufindtRNA parameters:       Relaxed (Int Cutoff= -32.1)
-------------------------------------------------------------

No tRNAs found.
```

Figura 22: Arquivo Texto Completo do Resultado de uma Tarefa

```
-------------------------------------------------------------
Sequence file(s) to search:  /mnt/win/phred/genome/workflow/seqs_fasta.contigs
Search Mode:                 Eukaryotic
Results written to:          Standard output
Output format:               Tabular
Searching with:              tRNAscan + EufindtRNA -> Cove
Covariance model:            TRNA2-euk.cm
tRNAscan parameters:         Strict
EufindtRNA parameters:       Relaxed (Int Cutoff= -32.1)
-------------------------------------------------------------

No tRNAs found.
```

Figura 23: Arquivo Texto Truncado do Resultado de uma Tarefa

Os exemplos *workflows* empregados nesta dissertação produzem arquivos de resultados com tamanho variando de poucos bytes até arquivos com 10 Kilobytes. Desta forma, foi empregado um método de compactação GZIP (P. Deutsch, 1996) para compactar alguns dos arquivos gerados na execução do *workflow* submetido. Este método pretende melhorar o desempenho do tempo de resposta do recebimento de resultados na interface dos dispositivos móveis. Foram considerados três tipos de tamanho de arquivos, sendo eles: Maior (maior que 6 Kylobytes), Médio (entre 3 e 6 Kylobytes) e Menor (menor que 3 Kylobytes). O gráfico 24 mostra a capacidade de compactação do método adotado e pode-se observar que os tipos de arquivos Maior, Médio e Menor alcançaram respectivamente uma redução de 66%, 77% e 53% em relação ao tamanho do arquivo normal.

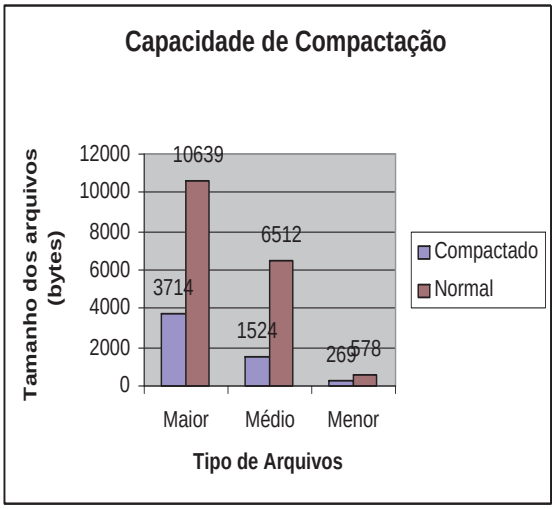

Figura 24: Capacidade de Compactação

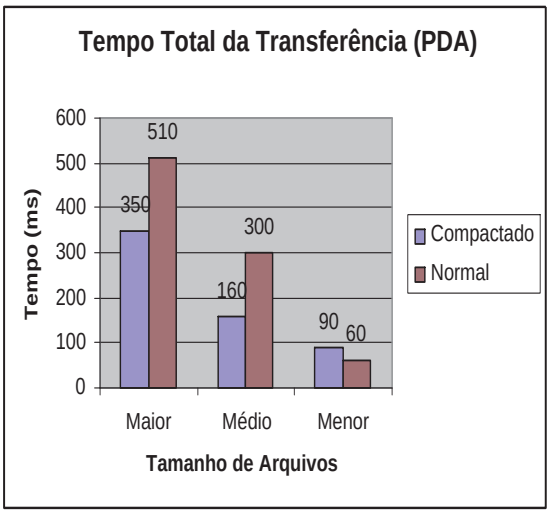

Figura 25: Tempo Total de Transferência

No entanto, quando analisando o tempo total de transferência do arquivo enviado (conforme

mostrado no gráfico 25), foi observado que somente os arquivos compactados com tamanho grande e médio apresentam menor tempo de resposta do que os respectivos arquivos normais ou sem compactação. Portanto, houve um melhor desempenho em arquivos que estão entre 6 e 10 Kilobytes usando o método de compactação. Por outro lado, no tipo de tamanho pequeno o recebimento do arquivo normal mostrou-se com um menor tempo de resposta. Desta forma, somente para arquivos que se enquadram entre a faixa de 3 Kylobytes até 10 Kylobytes foi empregado um método de compactação de arquivos, e abaixo de 3 Kylobytes não foi adotado método de compactação. Importante salientar que tempo total de transferências dos arquivos compactados mostrado no gráfico 25 contabiliza o tempo de compactação, tempo de transferência através das redes sem fio e estruturada e tempo de descompactação no dispositivo móvel.

A Figura 26 ilustra a descrição detalhada dos recursos (por exemplo, programas e banco de dados) utilizados nestes *workflows*, assim como a função que cada tarefa desempenha para resolver o problema representado pelo *workflow*. Esta descrição pode ser visualizada através da opção **Show Description**. Como ilustrado na Figura 27, possíveis problemas ou *warnings* gerados durante a execução de cada tarefa podem ser visualizados através da opção **Show Problems** do Menu. Por exemplo, formato do arquivo cromatograma está incorreto e a variável ambiente requisitada pelo programa não está configurada.

Alguns *workflows* podem levar até horas ou dias para terminar sua execução, assim como observado por (OINN et al., 2004). Entretanto, os dispositivos móveis possuem baterias com pequena capacidade de carga, não permitindo que estes aparelhos permaneçam ligados por um tempo suficiente para monitorar esses *workflows* que demandam muito tempo para executar. Desta forma, foi desenvolvida uma interface que permite que os usuários possam monitorar *workflows* que já concluiram sua execução ou estão ainda executando. Nesta interface o usuário deve especificar um valor numérico do Identificador de execução do pedido de submissão do *workflow* feito anteriormente para que possa ter as informações de status de cada tarefa atualizadas na interface de monitoramento. Além disso, o usuário pode normalmente visualizar os resultados e problemas da execução daquele *workflow* anteriormente submetido através das

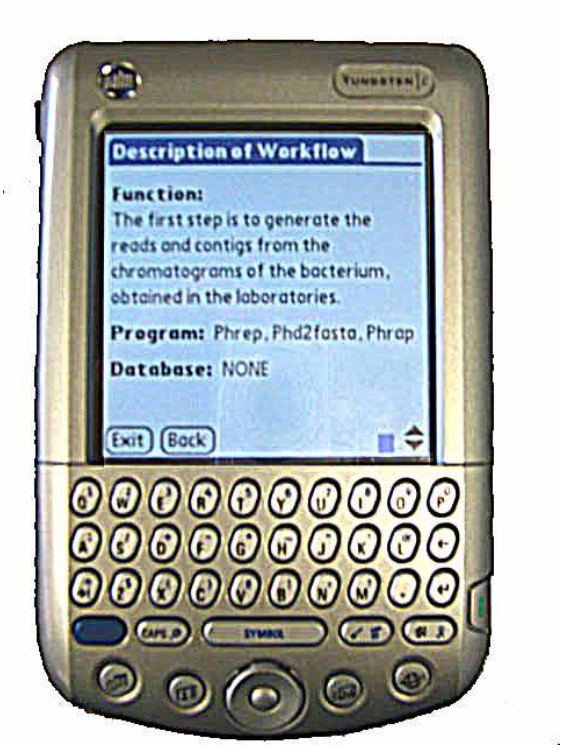

Figura 26: Descrição do Workflow

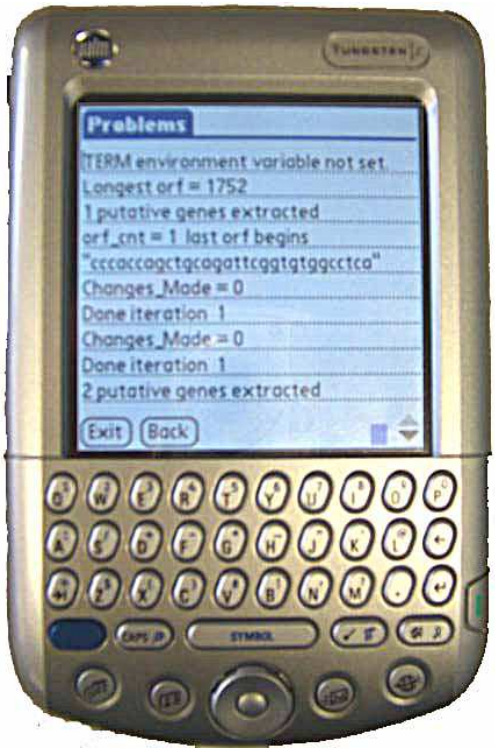

Figura 27: Problemas de uma tarefa

interfaces do Portal.

5.3.2 Interfaces nos Telefones Celulares

Todas as funcionalidades contidas na interface do PDA são também fornecidas na interface dos celulares. As figuras 28 e 29 mostram as opções contidas no menu do protótipo implementado.

Na figura 30, é mostrada a funcionalidade de visualização do tempo de execução das tarefas do *workflow*. Na figura 31 é ilustrado a descrição do *workflow* apresentado. Já na figura 32, é mostrado o resultado de uma das tarefas do *workflow*.

O celular empregado no presente trabalho utiliza a torre de transmissão de uma operadora de celular para transmitir dados na Internet utilizando a tecnologia de transmissão de dados GPRS (BETTSTETTER; VÖGEL; EBERSPÄCHER, 1999), que contabiliza uma tarifação por volume de dados transmitidos e em adição, o celular possui uma taxa de transmissão menor do que PDA (vide tabela 4). Desta forma, na interface dos telefones celulares foi empregada o mesmo

Figura 28: Primeira parte do Menu

Figura 29: Segunda Parte do Menu

Figura 30: Tempo

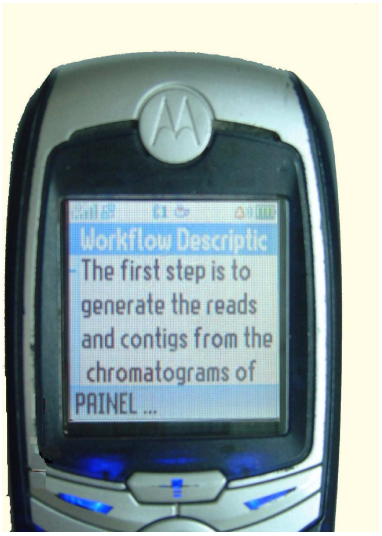

Figura 31: Descrição

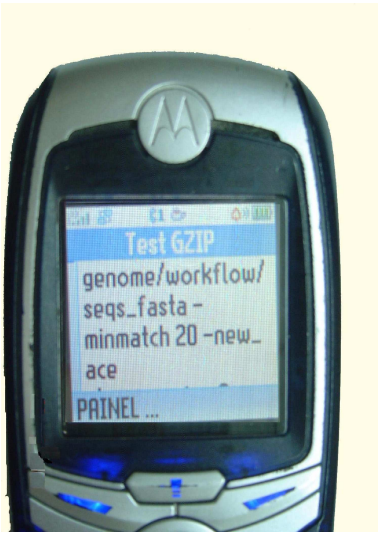

Figura 32: Resultado

método de compactação da interface PDA. O gráfico 33 ilustra a comparação entre o custo financeiro obtido com e sem utilizar o método de compactação. Pode-se notar uma economia de custo nos tamanhos de arquivos maior, médio e menor de respectivamente 65%, 77% e 34%. Como mostrado no gráfico 34, o tempo de transferência de arquivos mostrou-se menor quando utilizando a compactação em todos os tipos de arquivos. Desta forma, adotou-se o metódo de compactação na interface de telefones celulares em todos os tipos de arquivos.

A KVM do telefone celular utilizado nesta dissertação não forneceu suporte à API desenvolvida por (Robert Esser, 2006) e utilizada para desenhar a estrutura do *workflow* na interface

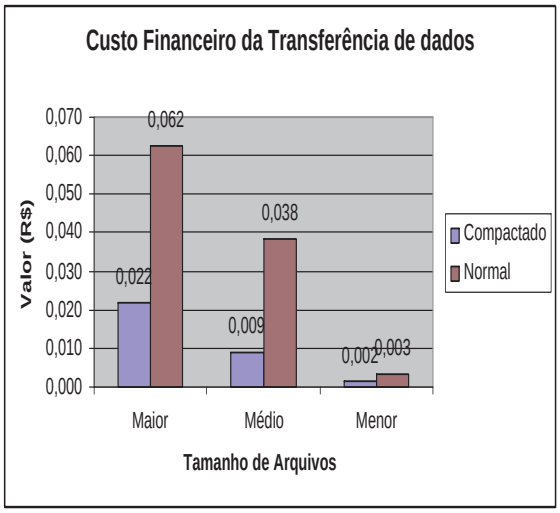

Figura 33: Custo Financeiro

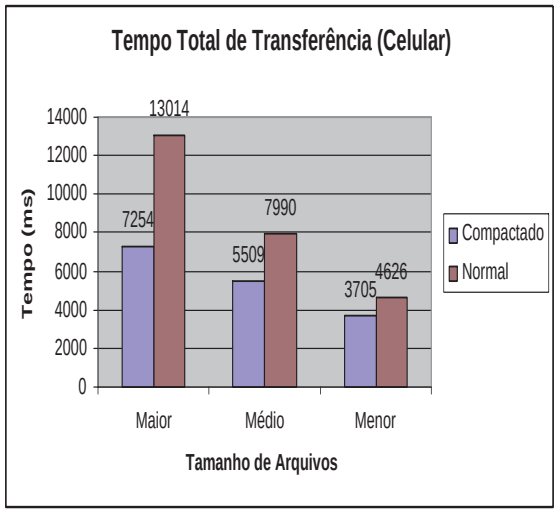

Figura 34: Tempo Total de Transferência

dos dispositivo móvel. Desta forma, foi necessário representar ou apresentar o status das tarefas *workflows* de outra maneira. Sendo assim, a interface de monitoramento desenvolvida para este tipo de dispositivo móvel apresenta o status das tarefas em um formulário com um campo de texto para cada tarefa, conforme mostrado nas figuras 35 e 36.

Figura 35: Interface de Monitoramento

Figura 36: Interface de Monitoramento

A interface de monitoramento nos telefones celulares é menos sofisticada que a interface de monitoramento do PDA, devido as restrições dos recursos deste aparelho estarem em um grau mais elevado do que os recursos encontrados PDAs. A interface dos celulares não possui a vivacidade fornecida pelas cores da interface PDA, mas também apresenta a funcionalidade de

monitorar de forma constantemente atualizada o conjunto de tarefas do *workflow* submetido.

5.4 Análise Comparativa da Abordagem Proposta

Como visto anteriormente na seção 4, as abordagens de submissão e monitoração de tarefas na maioria dos trabalhos apresentados tratam a submissão e monitoração de uma única tarefa por vez na interface do dispositivo móvel. No entanto, a utilização da abordagem de submissão de tarefas através do emprego de *workflow* permite a submissão de várias tarefas através de um único pedido de submissão. Isto é possível uma vez que o mecanismo *workflow* controla o fluxo de execução destas tarefas e invoca as ferramentas computacionais necessárias na execução das mesmas sem requisitar interação de usuários para realização de todos os passos. Na verdade, o usuário submete o *workflow* e as várias tarefas que fazem parte do mesmo são submetidas de forma automatizada e controlada por este mecanismo. Ao contrário do que acontece em (AHMAD et al., 2004; HWANG; ARAVAMUDHAM, 2004; CHU; HUMPHREY, 2004; GONZALEZ-CASTANO; VALES-ALONSO; LIVNY, 2003), onde cada tarefa é submetida por vez através da interface de usuário.

A submissão de várias tarefas através de um único pedido fornece uma melhor agilidade para resolver um único problema através da execução coordenada de várias tarefas empregando o mecanismo *workflow*, conforme pode ser visto no gráfico da Figura 37. Nesta análise foi realizado um comparativo entre a abordagem de submissão de tarefas de um exemplo *workflow* com 7 tarefas mostrado em (LEMOS, 2004) e o mesmo conjunto de tarefas sem o emprego de *workflow*. Na abordagem sem utilização do conceito *workflow*, as 7 tarefas foram submetidas pelo usuário que interage diretamente em cada submissão e controla a ordem de submissão dessas tarefas.

Foi realizado um total de 10 execuções a partir de um PDA para comparar o tempo de execução entre as duas abordagens. No caso da abordagem que não emprega *workflow*, o tempo total calculado em cada execução é o tempo somado de execução de cada tarefa. Já na abordagem utilizando *workflow*, o tempo total calculado trata-se do tempo total da execução do

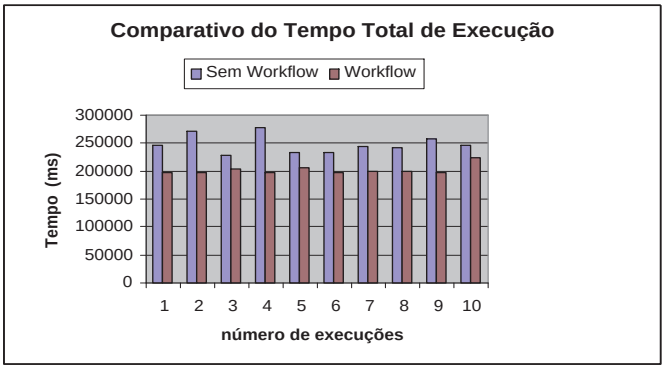

Figura 37: Comparativo do Tempo Total de Execução

mesmo. Nas duas abordagens, o tempo total de execução não contém o tempo gasto para o usuário interagir com interface do Portal. No caso da abordagem sem *workflow* se contabiliza o tempo a partir do momento que o usuário realizou a necessária interação com a interface para acontecer a submissão de cada uma das 7 tarefas. Já no caso da abordagem que utiliza *workflow* contabiliza o tempo a partir do momento que o usuário envia o pedido de submissão do *workflow*.

Importante salientar que os mesmos recursos de hardware e software (por exemplo, quantidade de memória, processador, ferramentas e sistema operacional) foram totalmente dedicados para a execução das duas abordagens. Os tempos totais de execução de todos os pedidos de submissão se mostraram menor na abordagem que utiliza *workflow*, o que é justificado pelo fato desta abordagem ser um procedimento automatizado, economizando o tempo gasto para processar os pedidos de envio e recebimento de cada tarefa e também ocorrendo menos acesso à rede sem fio, diferentemente da outra abordagem que não utiliza *workflow*.

Desta forma, o servidor processa menos pedidos de submissão e a rede sem fio trafega menos pedidos de submissão, o usuário precisa interagir menos para submissão de tarefas através do dispositivo móvel, resultando em uma redução do tempo total de execução para resolver um único problema e possibilitando ao usuário ocupar-se com outras atividades enquanto espera o término da execução do *workflow* (como por exemplo, visualizar resultados parciais do *workflow*). Outro aspecto positivo que a abordagem de *workflow* proporciona é a redução na utilização da largura de banda de rede sem fio, enviando menos pedidos de submissão.

Como a rede sem fio possui taxa de erros elevada comparada à rede estruturada em cabos, devido a este tipo de rede sofrer com interferência, ruídos e perturbações no canal (ECKHARDT; STEENKISTE, 1996), uma abordagem que envie muito menos pedidos de submissão de tarefas, como acontece no mecanismo *workflow*, reduz a chance de ocorrerem atrasos impostos pelo reenvio de alguns pacotes que possam ter sofrido algum erro enquanto trafegando pela rede sem fio.

Além disso, a redução de acesso a rede sem fio por parte dos dispositivos móveis proporciona uma menor dissipação de energia das baterias desses aparelhos (isto é, aumento do seu tempo de vida), como pode ser visto na gráfico 38. Prolongar o tempo de vida da bateria destes dispositivos tem sido um dos problemas mais críticos e desafiadores para os pesquisadores da computação móvel, como foi relatado em (RONG; PEDRAM, 2003; MOHAPATRA et al., 2005).

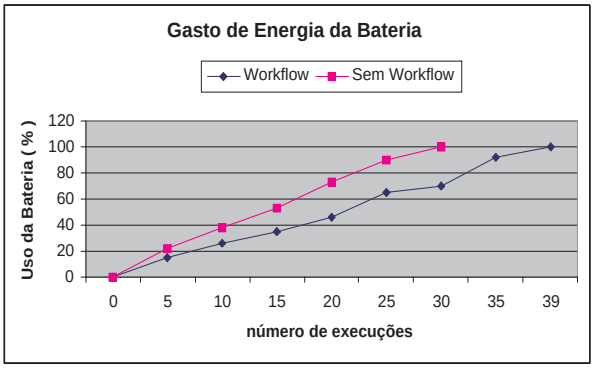

Figura 38: Uso da Bateria do Dispositivo Móvel

Semelhante ao experimento da Figura 37, no experimento da Figura 38 foi realizado uma comparação entre uma abordagem que utiliza *workflow* (isto é, coordenando e organizando as 7 tarefas que trabalham juntas para resolver o mesmo problema) e uma abordagem que não emprega *workflow*. Os resultados mostram que a abordagem *workflow* alcançou uma economia de 23% em relação à abordagem que não emprega *workflow*, permitindo um número maior de execuções (isto é, 9 execuções a mais). Para medir a porcetagem de bateria, foi utilizado o software BatteryGraph versão 1.21 (Jeroen Witteman, 2003), bastante empregados em PDAs com especificação Palm Tungsten. Além disso, este software permite visualizar a voltagem gasta durante os últimos dias através de um gráfico de linha e também outras informações que

permitem monitorar com mais facilidade a bateria.

5.5 Avaliação de Usabilidade

Com o intuito de fornecer uma melhor qualidade de interação das interfaces nos dispositivos móveis desenvolvidas na abordagem proposta nesta dissertação, foram realizados dois ciclos de avaliação de usabilidade. As etapas dos ciclos de avaliação são: preparação, realização e análise dos dados coletados (isto é, qualitativos e quantitativos). As principais referências utilizadas são: (CYBIS, 2002), (HIX; HARTSON, 1993), (NIELSEN, 1993) e (RUBIN, 1994).

Durante a etapa de preparação foram selecionadas as técnicas a serem empregadas, a elaboração dos roteiros de testes e questionários, a seleção de participantes e monitores para participar dos testes e, por fim, a escolha e preparação do cenário de teste. Dentre as técnicas estudadas, a avaliação realizada nesta dissertação empregou as técnicas de pesquisa de opinião e empírica. A técnica de pesquisa de opinião é baseada na aplicação de questionários ou entrevistas com o usuário para avaliar o seu grau de satisfação, utilização, funcionalidade, confortabilidade e aprendizagem em relação às interfaces. Já a técnica empírica conta com a participação direta dos usuários, utilizando experimentos que compreendem basicamente os testes com usuários através do monitoramento de sessões de uso (CYBIS, 2002).

Segundo (HIX; HARTSON, 1993), a técnica empírica é considerada a mais confiável, devido à complexidade e dificuldade de modelar o comportamento humano de forma geral. Uma solução que envolve a experimentação e observação do usuário realmente utilizando o produto permite uma melhor avaliação da usabilidade da interface. Desta forma, a empírica torna-se a mais ideal. Ainda na etapa de preparação, 5 usuários foram convidados para a realização dos testes, dentre eles 2 eram biólogos e 3 eram cientistas da computação. Vale a pena ressaltar a dificuldade e o transtorno em conseguir usuários para participar dos testes, visto que os testes demandam um certo período de tempo.

Na etapa de realização, pelo menos um especialista (isto é, um monitor) monitorava o usuário durante os testes, observando as dificuldades do usuário na utilização da interface e

também falhas da mesma. A parte final da etapa de realização consistia em preencher um questionário e um formulário de sugestões para a interface por parte do usuário.

Já na etapa de análise dos dados, assim como mostrada no gráfico da figura 39, a usabilidade da interface foi avaliada através da definição de seis variáveis principais, que são: facilidade (qual avaliação o usuário faz da facilidade de utilizar as interfaces), aprendizagem (que nível de aprendizagem as interfaces fornecem), satisfação (qual o grau de satisfação do usuário em utilizar as interfaces), utilidade (como o usuário avalia o grau de utilidade da interface), funcionalidade (a interface desempenhou a função que o usuário gostaria) e confortabilidade (as interfaces são confortáveis de utilizar). O questionário empregado nos testes pode ser encontrado no apêndice F.

A "Média Antes"e a "Média Depois"são referentes respectivamente à média dos questionários preenchidos na primeira e segunda versão das interfaces em relação a cada item ou variável. Na primeira versão, existia pouca preocupação com a usabilidade das interfaces. Já na segunda versão, foram empregadas melhorias detectadas a partir da avaliação da primeira versão. O peso trata-se de uma nota que o usuário fornece para cada um dos itens ou variáveis que fazem parte da avaliação da interface. Nos testes foram adotados pesos de 1 a 5 para todas variáveis da análise, sendo que 1 é o peso mais fraco e 5 identifica o peso mais forte.

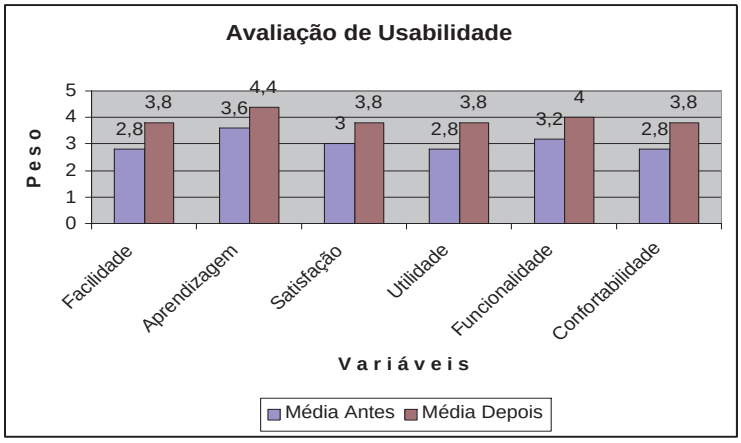

Figura 39: Avaliação de Usabilidade

Foram realizados testes e o preenchimento de questionários nas duas diferentes versões da interface. Na análise da primeira versão, foram detectados vários problemas que comprometiam

um pouco a utilização das interfaces por parte dos usuários, como pode ser visto na figura 39. Desta forma, foram estabelecidos graus de severidade para estes problemas e sugestões, e notou-se que uma parte dos problemas possuía grau de severidade 3 (Severo) e 4 (Inutilizado) em uma escala que vai de 0 a 4 (NIELSEN, 1993). Sendo assim, optou-se por refazer a avaliação em uma segunda versão, que propunha resolver todos os problemas detectados durantes a avaliação da primeira versão.

De acordo com a figura 39, pode-se observar que a segunda versão possui uma melhor média na avaliação dos usuários em todas as variáveis. Em outras palavras, a segunda versão teve melhor aceitação de forma geral. Fato que pode ser explicado pelas melhorias implantadas nesta versão que não existiam na versão anterior, ou seja, na primeira versão. Portanto, à medida que as melhorias foram identificadas e implantadas, as interfaces também melhoraram o seu grau de usabilidade, fornecendo, desta forma, interfaces mais amigáveis.

6 *Conclusões e Trabalhos Futuros*

Nesta dissertação foi abordado um problema específico na interação de grades computacionais e dispositivos móveis: a dificuldade de submeter e monitorar várias tarefas a partir de dispositivos móveis, onde estas tarefas trabalham cooperativamente e coordenadamente para alcançar um único objetivo que é resolver um problema específico. Desta forma, a abordagem adotada para solucionar esta dificuldade foi a utilização do mecanismo *workflow*.

Neste contexto, esta dissertação justificou a importância, levantou os requisitos e apresentou uma proposta de uma abordagem que permitisse a submissão, monitoração e obtenção de diversas outras informações sobre a execução de *workflows* em grades computacionais a partir de dispositivos móveis. A abordagem possui dois componentes. O primeiro componente é um Gerenciador de Workflow, que trata de receber requisições dos dispositivos móveis (isto é, requisições de submissão, monitoração e resultados de *workflows*) e interagir com as configurações de grade. Desta forma, este componente funciona como um intermediário entre o *middleware* de grade e os dispositivos móveis. Já o segundo componente é um Portal desenvolvido especificamente para dispositivos móveis (como por exemplo, celular e PDAs). Este componente fornece diversas interfaces que permitem ao usuário móvel submeter, monitorar, visualizar problemas e/ou resultados finais e parciais da execução de *workflows* necessários para resolver um problema específico.

Embora a princípio parecesse ser improvável a aplicação de *workflows* de configurações grade em dispositivos móveis, sua utilização foi verificada como um diferencial para a submissão de várias tarefas na configuração de grade e monitoração destas mesmas tarefas em dispositivos móveis de forma organizada, controlada e automatizada. Desta forma, conforme

mostrado ao longo das seções de análise comparativa e protótipo implementado, o emprego de *workflow* permitiu algumas vantagens que as abordagens anteriores não forneciam, como por exemplo: uma maior agilidade na submissão e monitoração de várias tarefas cooperantes, menor gasto de energia da bateria dos equipamentos móveis na submissão dessas tarefas e por último, minimizando a chance de ocorrer erros na ordenação de submissão das tarefas (isto é, uma vez que o processo se torna automatizado, o usuário não precisa interagir para que a submissão das tarefas ocorra, diminuindo assim os atrasos e erros por parte do usuário).

Através da pesquisa realizada com relação à aplicação de *workflows* de grade computacional em dispositivos móveis, podemos constatar que, apesar das restrições apresentadas nos dispositivos móveis, sua utilização pode ser viável em alguns casos. A solução desenvolvida oferece um ganho de produtividade a alguns profissionais fora de seus locais habituais de trabalho, através de um mecanismo que fornece também automatização, controle e organização. Entretanto, devido às restrições impostas a estes aparelhos, a interface contida nos mesmos apresenta certas limitações com relação a interfaces de computadores comuns. Um exemplo disso é o fato de não ser disponibilizada ainda a funcionalidade de composição *workflow* a partir do dispositivo móvel, devido principalmente às restrições de tela dos dispositivos móveis (por exemplo, tamanho pequeno e resolução baixa).

Desta forma, é importante ressaltar o esforço realizado para adaptar ou otimizar as interfaces do Portal e a abordagem *workflow* em conformidade com as restrições impostas pelos dispositivos móveis. Procurou-se desenvolver interfaces simples, estruturadas, padronizadas e que apresentassem informações de forma mais sucinta sem qualquer perda de consistência para o usuário. Os aspectos implementados que permitiram estas características são:

- Monitoramento *workflow* através de uma interface única e estruturada (isto é, organizada);
- Definição do intervalo de monitoração *workflow*, adequando-se à atual capacidade ou situação do dispositivo móvel;
- Voltar a monitorar ou ver como terminou um pedido de submissão de *workflow* realizado anteriormente;

- Otimização dos resultados, retirando trechos dos arquivos que não apresentavam relevância na interpretação dos resultados;
- Compressão ou compactação dos resultados no instante de transferir para os dispositivos móveis, resultando em um menor tempo de transferência para os dispositivos móveis, que no caso específico dos telefones celulares proporciona um menor custo financeiro da transferência dos arquivos;
- Emissão de sons na interface como forma de aviso. Por exemplo, quando o *workflow* termina execução é emitido um alerta em forma de som por certo período de tempo.

Para que fosse possível fornecer uma abordagem de *workflow* mais portável para os diferentes dispositivos (por exemplo, PDA e celulares) foi necessária uma investigação detalhada sobre quais tecnologias deveriam ser adotadas para desenvolver as interfaces do Portal e o componente Gerenciador de Workflow. A utilização das tecnologias J2ME, Java Servlet, HTTP e Java CoG Kit permitiu desenvolver os componentes e seus respectivos sub-componentes de forma satisfatória para alcançar este objetivo, atingindo uma grande diversidade de dispositivos móveis.

Entretanto, devido à existência de aparelhos que apresentam uma maior restrição de recursos (por exemplo, telefones celulares apresentando dispositivos de saída e entradas mais limitados que no PDA), foi necessário desenvolver interfaces adaptadas para cada tipo de dispositivo. A única diferença entre o Portal contido em um PDA e um Portal contido no telefone celular é a interface de monitoração de *workflow*, que se apresentou de maneira diferente em cada tipo de dispositivo. No PDA a interface representa graficamente a estrutura não-DAG especificada usando uma API do J2ME. No entanto, alguns modelos de celulares não possuem suporte a esta API. Desta forma, no celular foi adotado outro estilo de interface para visualização do status do *workflow*, sendo que esta interface possui ainda as características de ser única, estruturada e organizada, entretanto perdendo um pouco da característica de vivacidade ou animação.

A linguagem e o motor de execução *workflow* Karajan se mostraram realmente eficientes durante a definição e execução dos dois exemplos de *workflows* de bioinformática. Apesar

de ser uma linguagem recente, ela mostrou que possui uma sintaxe simples, clara e intuitiva, além de fornecer todas as estruturas comuns e várias outras estruturas avançadas que permite a definição de *workflows* complexos. Além disso, motor de execução Karajan é disponibilizado como uma API do Java CoG Kit que pemitem o desenvolvimento rápido e fácil de programas ou ferramentas que interagem com o *middleware* Globus. Sendo assim, o desenvolvimento do componente Gerenciador Workflow foi baseado nesta API.

Pode-se concluir que os objetos de pesquisa propostos inicialmente foram alcançados. Em outras palavras, as interfaces dos dispositivos móveis proporcionam a submissão e monitoração de várias tarefas que trabalham cooperativamente para resolver um único problema de maneira mais automatizada, coordenada e organizada através do mecanismo *workflow*. Em adição, as interfaces fornecem mais informações detalhadas e mais funcionalidades do que as interfaces dos trabalhos anteriores forneciam e também adequadas às restrições de cada tipo de dispositivo móvel. Desta forma, um usuário móvel utilizando o Portal foi capaz de acessar a descrição de *workflow*, submeter e monitorar *workflows* à configuração grade transparentemente, visualizar o tempo de execução de cada tarefa, assim como visualizar problemas e resultados finais e parciais de cada tarefa *workflow*.

Como trabalhos futuros, serão empregadas medidas para fornecer um comunicação mais segura (por exemplo, SSL) entre os dispositivos móveis e o componente Gerenciador de Workflow. Outro aspecto interessante será a elaboração de um mecanismo para tratar possíveis falhas durante a execução coordenada de várias tarefas *workflow* gerenciada pelo componente Gerenciador de Workflow. Um exemplo disto seria a decisão que deve ser tomada sobre um determinado pedido de submissão *workflow* quando ocorrer a desconexão do dispositivo móvel, devido à natureza de intermitência da rede sem fio.

Referências

AALST, Wil M. P. van der; BARROS, Alistair P.; HOFSTEDE, Arthur H. M. ter; KIEPUSZEWSKI, Bartek. Advanced workflow patterns. In: *CooplS '02: Proceedings of the 7th International Conference on Cooperative Information Systems*. London, UK: Springer-Verlag, 2000. p. 18–29. ISBN 3-540-41021-X.

AHMAD, Naveed; ALI, Arshad; ANJUM, Ashiq; AZIM, Tahir; BUNN, Julian J.; HASSAN, Ali; IKRAM, Ahsan; LINGEN, Frank van; MCCLATCHEY, Richard; NEWMAN, Harvey B.; STEENBERG, Conrad; THOMAS, Michael; WILLERS, Ian. Distributed analysis and load balancing system for grid enabled analysis on hand-held devices using multi-agents systems. *GCC 2004, Proceedings. LNCS 3251 Springer 2004*, p. 947–950, 2004.

ALLEN, R. *Workflow: An Introduction*. [S.l.]: L. Fischer (ed) The Workflow Handbook 2001, published by the Workflow Management Coalition (WfMC), 2001.

ALLIANCE, Globus. *About the Globus Alliance*. 2006. Http://www.globus.org/ , acessado em: 17/02/2006.

ALONSO, Gustavo; AGRAWAL, Divyakant; ABBADI, Amr El; MOHAN, C. Functionality and limitations of current workflow management systems. *IEEE Expert*, 1997.

AMIN, Kaizar; LASZEWSKI, Gregor von; HATEGAN, Mihael; ZALUZEC, Nestor J.; HAMPTON, Shawn; ROSSI, Albert. Gridant: A client-controllable grid workflow system. In: *HICSS*. [S.l.: s.n.], 2004. p. 210–219.

ANATEL. *Agência Nacional de Telecomunicações*. 2006. Disponível em: <http://www.anatel.gov.br/Tools/frame.asp?link=/biblioteca/releases/2006/release_19_06_2006mm.pdf e acessado em 03/07/2006>.

APACHE Ant. 2006. Disponível em: <http://ant.apache.org/ e acessado em 01/04/2006>.

BETTSTETTER, Christian; VÖGEL, Hans-Jörg; EBERSPÄCHER, Jörg. GSM phase 2+ general packet radio service GPRS: Architecture, protocols, and air interface. *IEEE Communication Surveys*, v. 2, n. 3, 1999.

BRAY, Tim; PAOLI, Jean; SPERBERG-MCQUEEN, C. M. *Extensible Markup Language (XML) 1.0 - W3C Recommendation*. [S.l.]: Disponível em: http://www.w3.org/TR/2004/REC-xml-20040204/, 1998.

BROOKE, John; PARKIN, Michael. A pda client for the computational grid. In: *WETICE '05: Proceedings of the 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise*. Washington, DC, USA: IEEE Computer Society, 2005. p. 325–330. ISBN 0-7695-2362-5.

BRUNEO, Dario; SCARPA, Marco; ZAIA, Angelo; PULIAFITO, Antonio. Communication paradigms for mobile grid users. *3rd (CCGrid'03)*, p. 669–676, May 2003.

BUYYA, R. *Grid Computing Info Centre (GRID Infoware)*. 2002. Http://www.gridcomputing.com/ , acessado em: 07/03/2006.

BUYYA, Rajkumar; VENUGOPAL, Srikumar. *The Gridbus Toolkit for Service Oriented Grid and Utility Computing: An Overview and Status Report*. 2004. Disponível em: <http://www.citebase.org/cgi-bin/citations?id=oai:arXiv.org:cs/0404027 e acessado em: 14/04/2006>.

CANNATARO, Mario; COMITO, Carmela; GUZZO, Antonella; VELTRI, Pierangelo. Integrating ontology and workflow in proteus, a grid-based problem solving environment for bioinformatics. In: *ITCC (2)*. [S.l.: s.n.], 2004. p. 90–94.

CASANOVA, H.; DONGARRA, J. *NetSolve: A Network Server for Solving Computational Science Problems*. Knoxville, TN, USA, 1995.

CHU, David C.; HUMPHREY, Marty. Mobile ogsi.net: Grid computing on mobile devices. In: *GRID '04: Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing (GRID'04)*. Washington, DC, USA: IEEE Computer Society, 2004. p. 182–191. ISBN 0-7695-2256-4.

CLARKE, B; HUMPHREY, M. Beyond the 'device as portal': Meeting the requirements of wireless and mobile devices in the legion grid computing system. *2nd International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing (associate with IPDPS 2002)*, p. 192–199, Abril 2002.

CURBERA, Francisco; DUFTLER, Matthew; KHALAF, Rania; NAGY, William; MUKHI, Nirmal; WEERAWARANA, Sanjiva. Unraveling the web services web: an introduction to soap, wsdl, and uddi. *IEEE Internet Computing*, v. 6, n. 2, p. 86 – 93, Mar. - Apr. 2002.

CYBIS, W. *Abordagem Ergonômica para IHC: Ergonomia de Interfaces Humano-Computador*. 2002. Disponível em: <www.labiutil.inf.ufsc.br/apostila/apostila.htm e acessado em 04/03/2006>.

DANTAS, Mario. *Computação Distribuída de Alto Desempenho, Redes, Clusters e Grids Computacionais*. [S.l.]: Axcel Books do Brasil Editora, 2005. ISBN 85-7323-169-6.

DEELMAN, Ewa; BLYTHE, James; GIL, Yolanda; KESSELMAN, Carl; MEHTA, Gaurang; PATIL, Sonal; SU, Mei-Hui; VAHI, Karan; LIVNY, Miron. Pegasus: Mapping scientific workflows onto the grid. In: *European Across Grids Conference*. [S.l.]: Springer, 2004. (Lecture Notes in Computer Science, v. 3165), p. 11–20. ISBN 3-540-22888-8.

DEELMAN, Ewa; KESSELMAN, Carl; MEHTA, Gaurang; MESHKAT, Leila; PEARLMAN, Laura; BLACKBURN, Kent; EHRENS, Phil; LAZZARINI, Albert; WILLIAMS, Roy; KORANDA, Scott. Griphyn and ligo, building a virtual data grid for gravitational wave scientists. In: *HPDC '02: Proceedings of the 11 th IEEE International Symposium on High Performance Distributed Computing HPDC-11 20002 (HPDC'02)*. Washington, DC, USA: IEEE Computer Society, 2002. p. 225. ISBN 0-7695-1686-6.

ECKHARDT, David; STEENKISTE, Peter. Measurement and analysis of the error characteristics of an in-building wireless network. In: *SIGCOMM '96: Conference proceedings on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM Press, 1996. p. 243–254. ISBN 0-89791-790-1.

ERWIN, Dietmar W.; SNELLING, David F. UNICORE: A Grid computing environment. *Lecture Notes in Computer Science*, v. 2150, p. 825–835, 2001.

FORMAN, George H.; ZAHORJAN, John. The challenges of mobile computing. *Computer*, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 27, n. 4, p. 38–47, 1994. ISSN 0018-9162.

FOSTER, Ian. Globus toolkit version 4: Software for service-oriented systems. *IFIP International Conference on Network and Parallel Computing*, p. 2–13, 2005.

FOSTER, Ian; GEISLER, J.; NICKLESS, W.; SMITH, W.; TUECKE, Steven. Software infrastructure for the i-way high performance distributed computing experiment. *5th IEEE Symposium on High Performance Distributed Computing*, p. 562–571, 1997.

FOSTER, Ian; KESSELMAN, Carl. *The Grid: Blueprint for a New Computing Infrastructure*. [S.l.]: Morgan Kaufmann, 1999. ISBN 1-55860-475-8.

FOSTER, Ian; KESSELMAN, Carl. *The Grid 2: Blueprint for a new Computing Infrastructure*. [S.l.]: John Wiley and Sons Ltd, 2003. ISBN 1-558-60933-4.

FOSTER, Ian; KESSELMAN, Carl; TUECKE, Steven. The anatomy of the grid: Enabling scalable virtual organizations. *International Journal of Supercomputer Applications*, v. 3, n. 15, 2001.

FOSTER, Ian; VÖCKLER, J; WILDE, M; ZHAO, Y. Chimera: A virtural data system for representing, querying, and automating data derivation. *14th International Conference on Scientific and Statistical DatabaseManagement (SSDBM)*, IEEE CS Press, 2002.

FRAUNHOFER RESOURCE GRID. 2006. Disponível em: <http://www.fhrg.fraunhofer.de/ e acessado em 14/04/2006>.

GLOBUS. *The Globus Toolkit*. 2006. Http://www.globus.org/ e acessado em: 02/03/2006.

GONZALEZ-CASTANO, F; VALES-ALONSO, J; LIVNY, M. Condor grid computing from mobile handheld devices. *ACM SIGMOBILE Mobile Computing and Communications Review*, v. 7, n. 1, p. 117–126, Janeiro 2003.

GREEN Group - University of Washington. 2006. Http://www.phrap.org/ , acessado em: 14/05/2006.

HIX, Deborah; HARTSON, H. Rex. *Developing user interfaces: ensuring usability through product & process*. New York, NY, USA: John Wiley & Sons, Inc., 1993. ISBN 0-471-57813-4.

HOHEISEL, Andreas. User tools and languages for graph-based grid workflows. *Special Issue of Concurrency and Computation:Practice and Experience*, 2005.

HOHEISEL, Andreas; DER, Uwe. An xml-based framework for loosely coupled applications on grid environments. In: *International Conference on Computational Science*. [S.l.]: Springer-Verlag, 2003. p. 245–254.

HOLLINGSWORTH, David. *The Workflow Reference Model*. [S.l.], 1995. Disponível em: <http://www.wfmc.org/standards/docs/tc003v11.pdf>.

HWANG, Junseok; ARAVAMUDHAM, Praveen. Middleware services for p2p computing in wireless grid networks. *IEEE Internet Computing*, IEEE Educational Activities Department, Piscataway, NJ, USA, v. 8, n. 4, p. 40–46, 2004. ISSN 1089-7801.

IBM. *IBM Grid*. 1998. Disponível em: <http://www.ibm.com/grid e acessado em 28/02/2006>.

IBM Software. *WebSphere Everyplace Micro Environment*. 2006. Disponível em: <http://www-306.ibm.com/software/wireless/weme/, acessado em: 26/05/2006>.

INNOVATION, IT. *IT Innovation home page*. 2006. Disponível em: <http://www.it-innovation.soton.ac.uk/ , acessado em: 20/03/2006>.

JENSEN, Kurt. An introduction to the theoretical aspects of coloured petri nets. In: *A Decade of Concurrency, Reflections and Perspectives, REX School/Symposium*. London, UK: Springer-Verlag, 1994. p. 230–272. ISBN 3-540-58043-3.

Jeroen Witteman. *BatteryGraph Overview*. 2003. Disponível em: <http://palm.jeroenwitteman.com/BatteryGraph/, acessado em: 19/10/2006>.

JOHNSON, A.; ROUSSOS, M.; LEIGH, J.; BARNES, C.; VASILAKIS, C.; MOHER, T. *The NICE project: Learning together in a virtual world*. 1998. Disponível em: <citeseer.ist.psu.edu/johnson98nice.html>.

JUNG, C; EINHOFF, M; NOLL, S; SCHIFFNER, N. Grid job builder - a workflow editor for computing grids. *(ITCC'04)*, v. 2, n. 2, p. 95–99, Abril 2004.

KRAVETS, R.; SCHWAN, K.; CALVERT, K. *Power-aware communication for mobile computers*. 1999.

KURKOVSKY, Stan; BHAGYAVATI, Arris Ray; YANG, Mei. Modeling a grid-based problem solving environment for mobile devices. *(ITCC'04)*, v. 2, n. 2, p. 135–136, Abril 2004.

LASZEWSKI, Gregor; HATEGAN, Mike. Workflow concepts of the java cog kit. *Journal of Grid Computing*, Springer Netherlands, v. 3, n. 3-4, p. 239–259, 2005. ISSN 0010-4620.

LASZEWSKI, G Von; FOSTER, Ian; GAWOR, J; LANE, P. A java commodity grid kit. *Concurrency and Computation: Practice and Experience*, v. 13, n. 8-9, p. 643–662, 2001.

LASZEWSKI, Gregor Von; HATEGAN, Mike. *Grid Workflow - An Integrated Approach*. Argonne, IL, USA, 2005. Disponível em: <Disponível em: http://www.mcs.anl.gov/ gregor/papers/vonLaszewski-workflow-draft.pdf e acessado em 13/04/2006>.

LEE, Craig A.; KESSELMAN, Carl; SCHWAB, Stephen. Near-real-time satellite image processing: Metacomputing in cc++. *IEEE Comput. Graph. Appl.*, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 16, n. 4, p. 79–84, 1996. ISSN 0272-1716.

LEMOS, Melissa. *Workflow para Bioinformática*. Dissertação (Mestrado) — Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), 2004. Disponível em: <www.inf.puc-rio.br/ melissa/publicacao/download/tese_melissa/Tese_Melissa_Lemos.pdf>.

LITKE, A.; SKOUTAS, D.; VARVARIGOU, T. Mobile grid computing: Changes and challenges of resource management in a mobile grid environment. *PAKM 2004 Conference*, 2004.

MCKNIGHT, Lee W.; HOWISON, James; BRADNER, Scott. Guest editors' introduction: Wireless grids–distributed resource sharing by mobile, nomadic, and fixed devices. *IEEE Internet Computing*, v. 8, n. 4, p. 24–31, 2004.

Microsoft Corporation 2006. *Microsoft .NET Homepage*. 2006. Disponível em: <http://www.microsoft.com/net/, acessado em: 26/05/2006>.

MICROSYSTEMS, Sun. *PersonalJava Application Environment*. 2006. Disponível em: <http://java.sun.com/products/personaljava/, acessado em: 22/04/2006>.

MOHAPATRA, Shivajit; CORNEA, Radu; OH, Hyunok; LEE, Kyoungwoo; KIM, Minyoung; DUTT, Nikil D.; GUPTA, Rajesh; NICOLAU, Alexandru; SHUKLA, Sandeep K.; VENKATASUBRAMANIAN, Nalini. A cross-layer approach for power-performance optimization in distributed mobile systems. In: *IPDPS*. [S.l.: s.n.], 2005.

MOORE, Reagan W.; BARU, Chaitanya; MARCIANO, Richard; RAJASEKAR, Arcot; WAN, Michael. Data-intensive computing. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, p. 105–129, 1999.

MYGRID, FreeFluo e. *Freefluo - The Workflow Enactment Engine*. 2006. Http://www.mygrid.org.uk/ e acessado em: 28/03/2006.

NIELSEN, J. *Usability Engineering*. Chestnut Hill, MA: Academic Press, 1993.

OINN, Tom; ADDIS, Matthew; FERRIS, Justin; MARVIN, Darren; SENGER, Martin; GREENWOOD, Mark; CARVER, Tim; GLOVER, Kevin; POCOCK, Matthew R.; WIPAT, Anil; LI, Peter. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, Oxford, UK, v. 20, n. 17, p. 3045–3054, 2004. ISSN 1367-4803.

P. Deutsch. *RFC 1952 - GZIP File Format Specification version 4.3*. 1996. Disponível em: <http://www.rfc-archive.org/getrfc.php?rfc=1952, acessado em: 23/06/2006>.

PARK, S.-M.; KO, Young-Bae; KIM, Jai-Hoon. Disconnected operation service in mobile grid computing. In: *ICSOC*. [S.l.: s.n.], 2003. p. 499–513.

PHAM, T; HUANG, L; DULAN, C. Challenge: Integrating mobile wireless devices into the computational grid. *8th (MOBICOM'02)*, p. 271–278, Setembro 2002.

Robert Esser. *Petri Net Browser*. 2006. Disponível em: <http://www.cs.adelaide.edu.au/ esser/browser.html, acessado em: 22/06/2006>.

RONG, Peng; PEDRAM, Massoud. Extending the lifetime of a network of battery-powered mobile devices by remote processing: a markovian decision-based approach. In: *DAC '03: Proceedings of the 40th conference on Design automation*. New York, NY, USA: ACM Press, 2003. p. 906–911. ISBN 1-58113-688-9.

RUBIN, Jeffrey. *Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests*. New York, NY, USA: John Wiley & Sons, Inc., 1994. ISBN 0471594032.

SAIRAMESH, J.; GOH, S.; STANOI, I.; PADMANABHAN, S.; LI, C. S. Disconnected processes, mechanisms and architecture for mobile e-business. *Mobile Network Application*, Kluwer Academic Publishers, Hingham, MA, USA, v. 9, n. 6, p. 651–662, 2004. ISSN 1383-469X.

SHAHAB, Atif; CHUON, Danny; SUZUMURA, Toyotaro; LI, Wilfred W.; BYRNES, Robert W.; TANAKA, Kouji; ANG, Larry; MATSUOKA, Satoshi; BOURNE, Philip E.; MILLER, Mark A.; ARZBERGER, Peter W. Grid portal interface for interactive use and monitoring of high-throughput proteome annotation. In: *LSGRID*. [S.l.: s.n.], 2004. p. 53–67.

SKILLICORN, D. B. Motivating computational grids. In: *CCGRID '02: Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid*. Washington, DC, USA: IEEE Computer Society, 2002. p. 401. ISBN 0-7695-1582-7.

SOTOMAYOR, Borja. *The Globus Toolkit version 3 Programmer's Tutorial*. 2003. Http://gdp.globus.org/gt3tutorial/singlehtml/progtutorial_0.4.3.html, acessado em: 12/04/2006.

SOTOMAYOR, Borja. *The Globus Toolkit 4 Programmer's Tutorial*. 2005. Http://gdp.globus.org/gt4tutorial/ e acessado em 12/04/2006.

SPOONER, D. P.; CAO, J.; JARVIS, S. A.; HE, L.; NUDD, G. R. Performance-aware workflow management for grid computing. *Computer Journal*, Oxford University Press, Oxford, UK, v. 48, n. 3, p. 347–357, 2005. ISSN 0010-4620.

STEVENS, R D; ROBINSON, A J; GOBLE, C A. mygrid:personalized bioinformatics on the information grid. *Bioinformatics*, Oxford University Press, v. 3, n. 3-4, p. 302–304, 2003.

Sun Microsystems. *J2EE(TM) 1.4 Tutorial*. 2006. Disponível em: <http://java.sun.com/j2ee/1.4/docs/tutorial/doc/, acessado em: 13/06/2006>.

Sun Microsystems. *Java Platform, Micro Edition (Java ME)*. 2006. Disponível em: <http://java.sun.com/javame/index.jsp, acessado em: 22/04/2006>.

Sun Microsystems. *Java Platform, Standart Edition (Java SE)*. 2006. Disponível em: <http://java.sun.com/javase/index.jsp, acessado em: 13/05/2006>.

TALIA, Domenico. The open grid services architecture: Where the grid meets the web. *IEEE Internet Computing*, IEEE Educational Activities Department, Piscataway, NJ, USA, v. 6, n. 6, p. 67–71, 2002. ISSN 1089-7801.

TANNENBAUM, Todd; WRIGHT, Derek; MILLER, Karen; LIVNY, Miron. Condor – a distributed job scheduler. In: STERLING, Thomas (Ed.). *Beowulf Cluster Computing with Linux*. [S.l.]: MIT Press, 2001.

TAYLOR, Ian; WANG, Ian; SHIELDS, Matthew; MAJITHIA, Shalil. Distributed computing with Triana on the Grid. *Concurrency and Computation:Practice and Experience*, To be published Wiley Publishing, v. 17, n. 1–18, 2005.

THAIN, Douglas; TANNENBAUM, Todd; LIVNY, Miron. Distributed computing in practice: the condor experience. *Concurrency - Practice and Experience*, v. 17, n. 2-4, p. 323–356, 2005.

TOPLEY, Kim. *J2ME in a Nutshell*. [S.l.]: O'Reilly, 2002. ISBN 0-596-00253-X.

TUECKE, S.; CZAJKOWSKI, K.; FOSTER, I.; REY, J.; STEVE, F.; CARL, G. *Grid Service Specification*. 2002. Disponível em: <http://www.globus.org/toolkit/draft-ggf-ogsi-gridservice-33_2003-06-27.pdf e acessado em 02/03/2006>.

UNICORE at SourceForge: Detailed AJO description. 2006. Disponível em: <http://unicore.sourceforge.net/ajo.html, acessado em: 22/04/2006>.

WANG, Minhong; WANG, Huaiqing. Intelligent agent supported business process management. In: *HICSS*. [S.l.: s.n.], 2005. p. 71b–71b.

WASSON, Glenn S.; BEEKWILDER, Norm; MORGAN, Mark M.; HUMPHREY, Marty. Ogsi.net: Ogsi-compliance on the .net framework. In: *CCGRID*. [S.l.: s.n.], 2004. p. 648–655.

WFMC. *Workflow Management Coalition Terminology and Glossar*. Brussels, 1999.

WIKIPéDIA. *DIRECTED ACYCLIC GRAPH*. 2006. Disponível em: <http://en.wikipedia.org/wiki/Directed_acyclic_graph e acessado em 14/04/2006>.

WIKIPéDIA. *Sistemas Cooperativos e de Workflow*. 2006. Disponível em: <http://pt.wikipedia.org/wiki/Sistemas_Cooperativos_e_de_WorkFlow e acessado em 15/03/2006>.

WSRF. *Globus: WS Resource Framework*. 2004. Disponível em: <http://www.globus.org/wsrf/ e acessado em 12/04/2006>.

YANG, M; LIANG, H; XU, B. S-wfms: A service-based workflow management system in grid environment. *19th (AINA'05)*, v. 1, n. 1, p. 293–297, March 2005.

YU, J; BUYYA, R. A taxonomy of workflow management systems for grid computing. *(SIGMOD'05)*, Chicago, IL, USA, v. 34, n. 3, p. 44–49, July 2005.

ZHOU, Songnian; ZHENG, Xiaohu; WANG, Jingwen; DELISLE, Pierre. Utopia: a load sharing facility for large, heterogeneous distributed computer systems. *Softw. Pract. Exper.*, John Wiley & Sons, Inc., New York, NY, USA, v. 23, n. 12, p. 1305–1336, 1993. ISSN 0038-0644.

APÊNDICE A – Publicações

A.1 Trabalhos Publicados

Título: Submissão e Monitoração Utilizando Dispositivos Móveis para a Abordagem de Gerenciamento de Workflow em Ambientes de Grades

Evento: VI WSCAD - Workshop em Sistemas Computacionais de Alto Desempenho

Local: Rio de Janeiro - RJ

Data: Outubro de 2005

Autores: Vinicius da Cunha Martins Borges e Mario Antonio Ribeiro Dantas

Título: Utilização de Dispositivos Móveis e Gerenciamento de Workflow para Submissão e Monitoração de Tarefas em Ambientes de Grids

Evento: ERAD - Escola Regional de Alto Desempenho

Local: Ijuí - RS

Data: Janeiro de 2006

Autores: Vinicius da Cunha Martins Borges e Mario Antonio Ribeiro Dantas

Título: Utilização de Dispositivos Móveis e Gerenciamento de Workflow para Submissão e Monitoração de Tarefas em Ambientes de Grids

Evento: *4th I2TS - International Information and Telecommunication Technologies Symposium*

Local: Florianópolis - SC

Data: Dezembro de 2005

Autores: Vinicius da Cunha Martins Borges e Mario Antonio Ribeiro Dantas

Título: Uma Abordagem de Submissão e Monitoração de Múltiplas Tarefas para Ambientes de Grade Computacional Utilizando Dispositivos Móveis

Evento: XXXIII SEMISH - Seminário Integrado de Software e Hardware

Local: Campo Grande - MS

Data: Julho de 2006

Autores: Vinicius da Cunha Martins Borges e Mario Antonio Ribeiro Dantas

APÊNDICE B – Configuração do Ambiente de Desenvolvimento

Nesta seção serão apresentados alguns dos programas utilizados para configuração do ambiente de desenvolvimento

B.1 Ambiente de Software

B.1.1 *Java 2 Micro Edition* (J2ME)

Java 2 Micro Edition fornece um ambiente para o desenvolvimento em dispositivos com memória, tela e poder de processamento limitados. J2ME inclui a máquina virtual java e um conjunto de APIs padrão do Java definidos através da Java Community.

A arquitetura J2ME define Configurações, Perfis e APIs opcionais, como ilustrado na Figura 40. Esta divisão permite ao desenvolvedor conhecer informações específicas sobre diferentes dispositivos. Uma configuração de J2ME é composta de uma máquina virtual e um conjunto reduzido de bibliotecas. Estas bibliotecas fornecem as funcionalidades básicas para uma grande variedade de dispositivos que compartilham características similares. Entre as configurações estão:

- **CLDC**: consiste de uma maquina virtual reduzida (KVM) e um conjunto de classes mais apropriadas para dispositivos de conexões de rede intermitentes, processadores lentos e memória limitada. Incorpora também um garbage collector que é otimizado para um ambiente de memória reduzida. KVM é comumente sendo usado em diversas implementações, mas existem outras máquinas virtuais que poderiam ser utilizadas, tais como J9

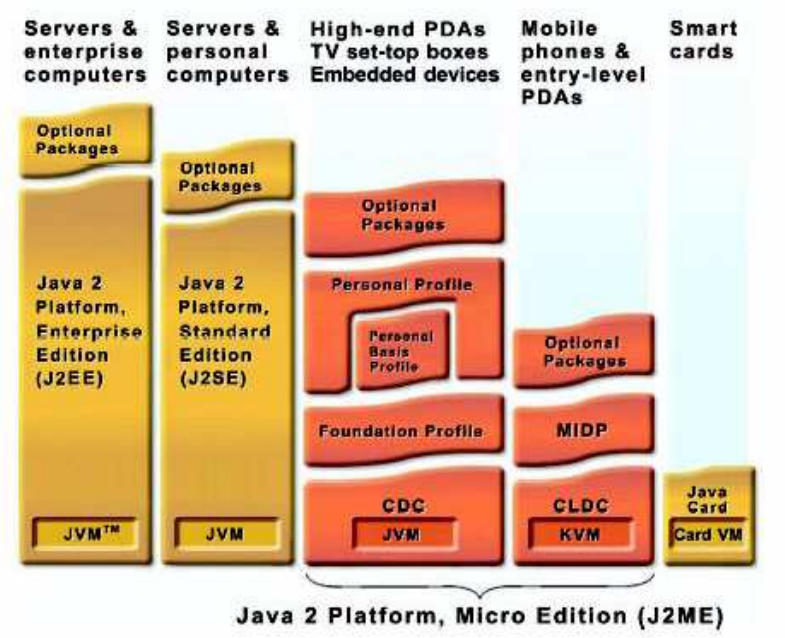

Figura 40: A arquitetura do J2ME (Sun Microsystems, 2006b)

VM da IBM. Nesta dissertação adotou-se o uso da configuração CLDC 1.1, devido ser a configuração que mais se adapta as características de hardware dos dispositivos móveis usados;

- **CDC**: discute a necessidade de dispositivos que se posicionam entre aqueles discutidos pelo CLDC e os completos sistemas desktops executando J2SE. Estes dispositivos tem mais memórias (tipicamente 2 MB ou mais) e processadores com mais capacidade, e eles podem, então, suportar um ambiente mais completo do Java. CDC pode ser encontrado em PDAs de alta capacidade, smart phones, telefones web, gateways residenciais e set-top boxes.

Para o fornecimento de um ambiente mais completo de execução para diferentes categorias de dispositivos, foi combinado um conjunto de APIs, ou perfis que definem o ciclo de vida da aplicação, a interface com o usuário, e acesso a características específicas dos dispositivos, conforme descrito abaixo:

- **Mobile Information Device Profile (MIDP)**: O perfil MIDP fornece funcionalidades requeridas pelas aplicações movéis incluindo interface com o usuário, acesso a rede e

armazenamento local. Combinado com CLDC, o MIDP fornece um completo ambiente de execução java e eleva as capacidades dos dispositivos móveis e minimiza o consumo de memória e energia;

- **Foundation Profile (FP)**: é um conjunto de APIs Java da configuração CDC que suporta dispositivos com recursos limitados sem um sistema GUI; ** Personal Profile (PP): é um grande conjunto de perfis da configuração CDC que suportam dispositivos com recursos limitados com uma GUI completa baseada em AWT;

- **Personal Basis Profile (PBP)**: é um subconjunto do PP, fornecendo um ambiente de aplicação para conectar a rede de dispositivos que suportem níveis básicos de representação gráfica ou que requeiram a utilização de ferramentas gráficas especializadas para determinadas aplicações.

A plataforma J2ME permite a combinação de vários pacotes com o CLDC, CDC e seus correspondentes perfis. As APIs opcionais permitem a utilização de tecnologias como, bluetooth, web services, wireless messaging, multimídia, e conexão com o banco de dados.

Nesta dissertação, para o desenvolvimento da aplicação no cliente móvel optou-se pela utilização do MIDP 2.0. Esta escolha foi realizada pelo fato desta versão do MIDP fornecer suporte a uma maior variedade de protocolos de comunicação, como pode ser visto na seção B.1.2.

B.1.2 Protocolos de Comunicação

A versão 1.0 do MIDP não suporta rede de baixo nível para sockets TCP/IP. Esta deficiência foi suprida na especificação MIDP 2.0, como resposta às necessidades das redes de computadores. Este suporte esta baseado no *Generic Connection Framework* (GCF) do CLDC. GCF fornece uma interface genérica para diferentes formas de comunicação (por exemplo, socket, datagrama e http) e URL padronizadas para indicar o tipo de conexão criada, como pode ser visto na tabela 5.

Tabela 5: Algumas formas de conexão

Tipo de Conexão	Método utilizado	Exemplo
HTTP	Connector.open	Connector.open("http://java.sun.com/wireless")
Socket	Connector.open	Connector.open("socket://java.sun.com:port")
Datagrama	Connector.open	Connector.open("datagram://java.sun.com:port")

B.1.3 Máquina Virtual do Dispositivo Móvel

WebSphere Everyplace Micro Environment (IBM Software, 2006) é uma ambiente de execução que fornece fundamentos para o desenvolvimento de aplicações em pequenos dispositivos móveis. Este ambiente contém em seu núcleo a máquina virtual J9 da IBM que suporta aplicações J2ME em vários dispositivos (por exemplo, PDA, telefones celulares e outros dispositivos embarcados), diferentes sistemas operacionais (por exemplo, PalmOS, PocketPC, WinCE, QNX, OSE, Linux e outros) e com diferentes perfis e configurações (por exemplo, MIDP, CLDC e CDC). Desta forma, optou-se por este ambiente de execução nos PDAs utilizados nesta dissertação, devido a portabilidade oferecida por este ambiente.

B.1.4 *Java 2 Standard Edition* (J2SE)

Existem dois produtos principais na família J2SE. *Java 2 Runtime Environment, Standard Edition (JRE)* e *Java 2 Software Development Kit, Standard Edition* (SDK). O JRE fornece APIs, a máquina virtual, e outros componentes necessários na execução de aplicações escritas na linguagem de programação java. O Java 2 SDK contém ferramentas como compiladores e debuggers necessários para o desenvolvimento de aplicações. Nesta dissertação foi utilizada a versão 1.5.0_04-b05 do J2SE.

B.1.5 *J2EE Aplicações Web - Servlet*

Servlet é uma tecnologia Java para extensão e melhoria de servidores *Web*. Esta tecnologia fornece uma forma independente de plataforma e baseada em componentes para o desenvolvimento de aplicações *Web* (Sun Microsystems, 2006a). Além disso, *servlets* têm acesso a toda família de APIs java, inclusive uma biblioteca de chamadas específicas do protocolo HTTP. Ela

fornece um framework geral para construir serviços usando o paradigma requisição-resposta (em inglês, *request-response*). Embora todos os *servlets* serem desenvolvidos em Java, os clientes podem ser desenvolvidos em qualquer linguagem (por exemplo, J2ME, applet, navegadores Web, aplicações clientes com GUI, PHP e .NET), como mostrado na figura 41.

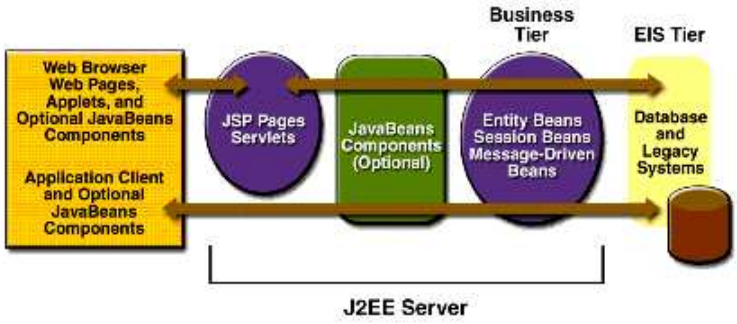

Figura 41: Componentes do J2EE (Sun Microsystems, 2006a)

Além disso, quando *servlets* são usados na camada do meio de aplicações de sistemas distribuídos (isto é, na camada *J2EE Server*), elas podem se tornar clientes de outros serviços ou sistemas legados. Como por exemplo, nesta dissertação o *servlet* comunica com sistema legado na camada EIS (*Enterprise Information System*) que trata-se de uma aplicação desenvolvida em *Java CoG Kit* para interagir com *middleware* Globus. Desta forma, este *servlet* desenvolvido nesta dissertação comunica com sistemas que invocam outras APIs alternativas do *Java CoG Kit*, como às vezes requisitados por este sistema. Na presente dissertação foi utilizada a especificação 2.4 da tecnologia *servlet*. Os motivos que levaram escolher esta especificação foram às melhorias acrescentadas nesta versão, as principais melhorias são: modularização do formato de desenvolvimento, melhor modelo de segurança, adição de pedido e resposta ao nível *listener* e notificações de evento e por último permite containers utilizar os novos pacotes J2SE de entrada e saída. Nesta dissertação não foi utilizado qualquer componente *JavaBeans*, ou seja, o *servlet* aqui desenvolvido comunica diretamente com a aplicação desenvolvido em *Java CoG Kit* contida na camada EIS.

Nesta dissertação, foi utilizado a versão estável do Apache tomcat 5.5.17 como sendo o *container servlet*, devido este *container* ser utilizado na implementação oficial para tecnologia *Java Servlet* e também por ser desenvolvido como um projeto aberto e gratuito dentro do projeto

Jakarta na *Apache Software Foundation*. Os motivos que levaram a escolher esta versão foram às melhorias fornecidas nesta versão que em outras versões anteriores não era fornecida. As principais melhorias são: otimizações de desempenho, redução de *garbage collection*, melhoramento de confiabilidade e escalabilidade e melhor integração de plataformas (isto é, Unix e Windows).

Quando um cliente conecta ao servidor e faz uma requisição HTTP, o pedido pode ser feito de várias formas ou métodos diferentes. Os métodos mais freqüentemente usados são GET e POST. Dos vários métodos definidos para este protocolo, MIDP suporta apenas GET e POST. A figura 42 ilustra de melhor forma o envio de uma requisição HTTP.

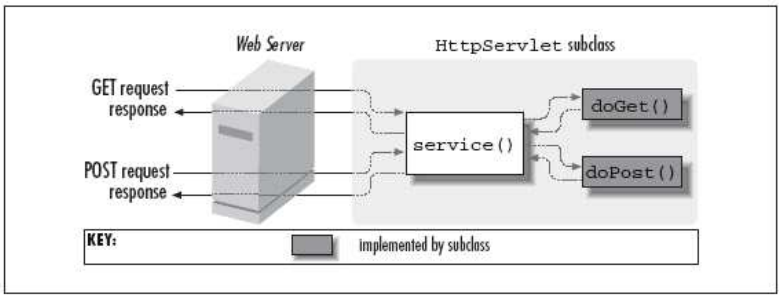

Figura 42: Requisição e Resposta HTTP (TOPLEY, 2002)

Ambos os métodos passam informações através da URL de conexão e de vários cabeçalhos. O que difere ambos é a maneira como dados complementares são enviados. No método GET, dados são passados como parâmetros na URL. No método POST eles são enviados através de uma stream de saída, que é criada depois de os cabeçalhos terem sido definidos. Como existe uma restrição para a quantidade de caracteres na URL, é interessante adotar como padrão o método POST. Além disso, a passagem de conteúdo binário só é possível através deste método. Por estas razões foi adotado o método POST para enviar dados para o *servlet* residente no servidor Web.

APÊNDICE C – Modelagem UML

A figura 43 apresenta os principais métodos e classes desenvolvidos no módulo Gerenciador Workflow.

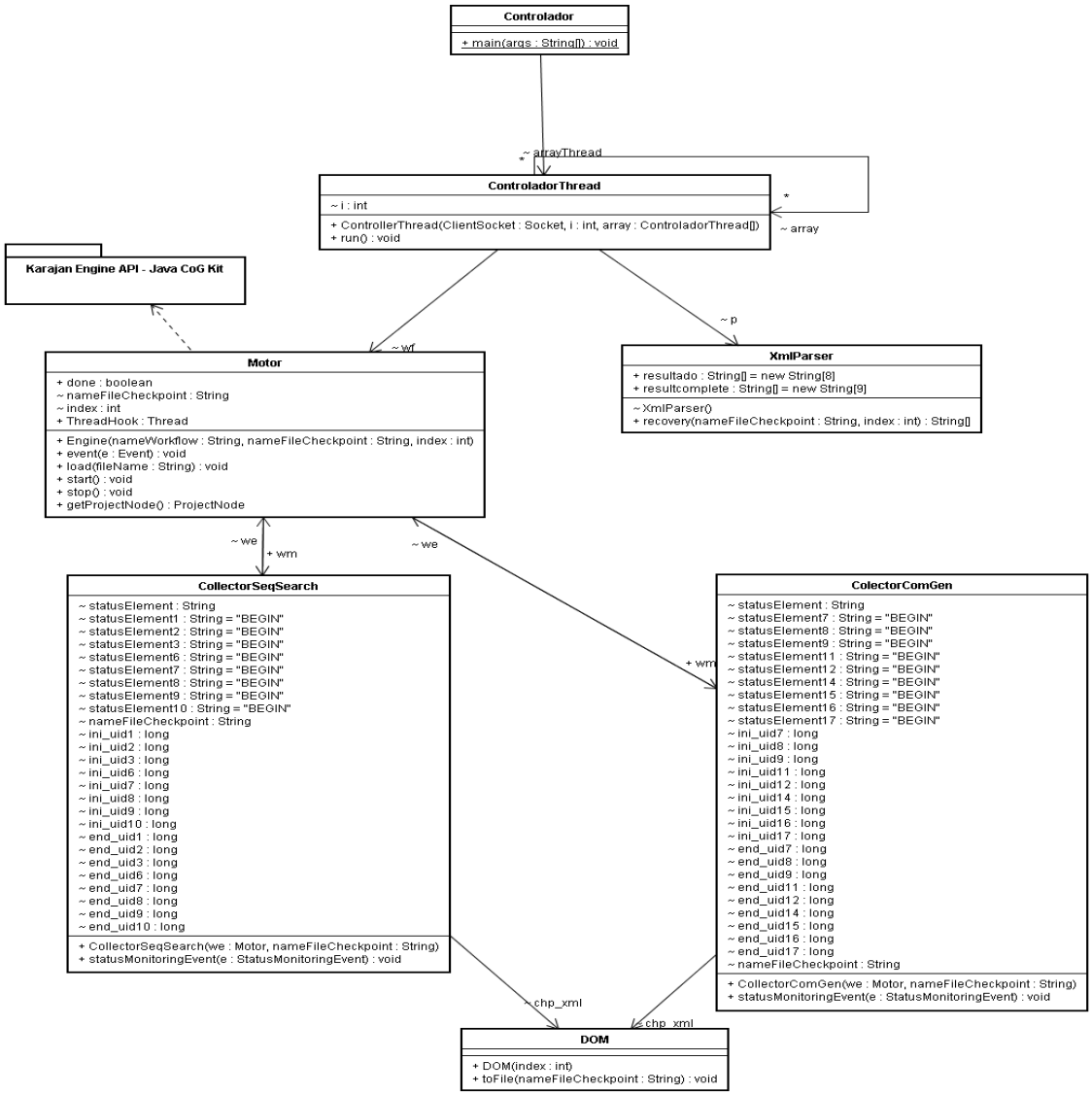

Figura 43: Diagrama de Classes do Gerenciador Workflow

A figura 44 apresenta os principais métodos e classes desenvolvidos no módulo Portal.

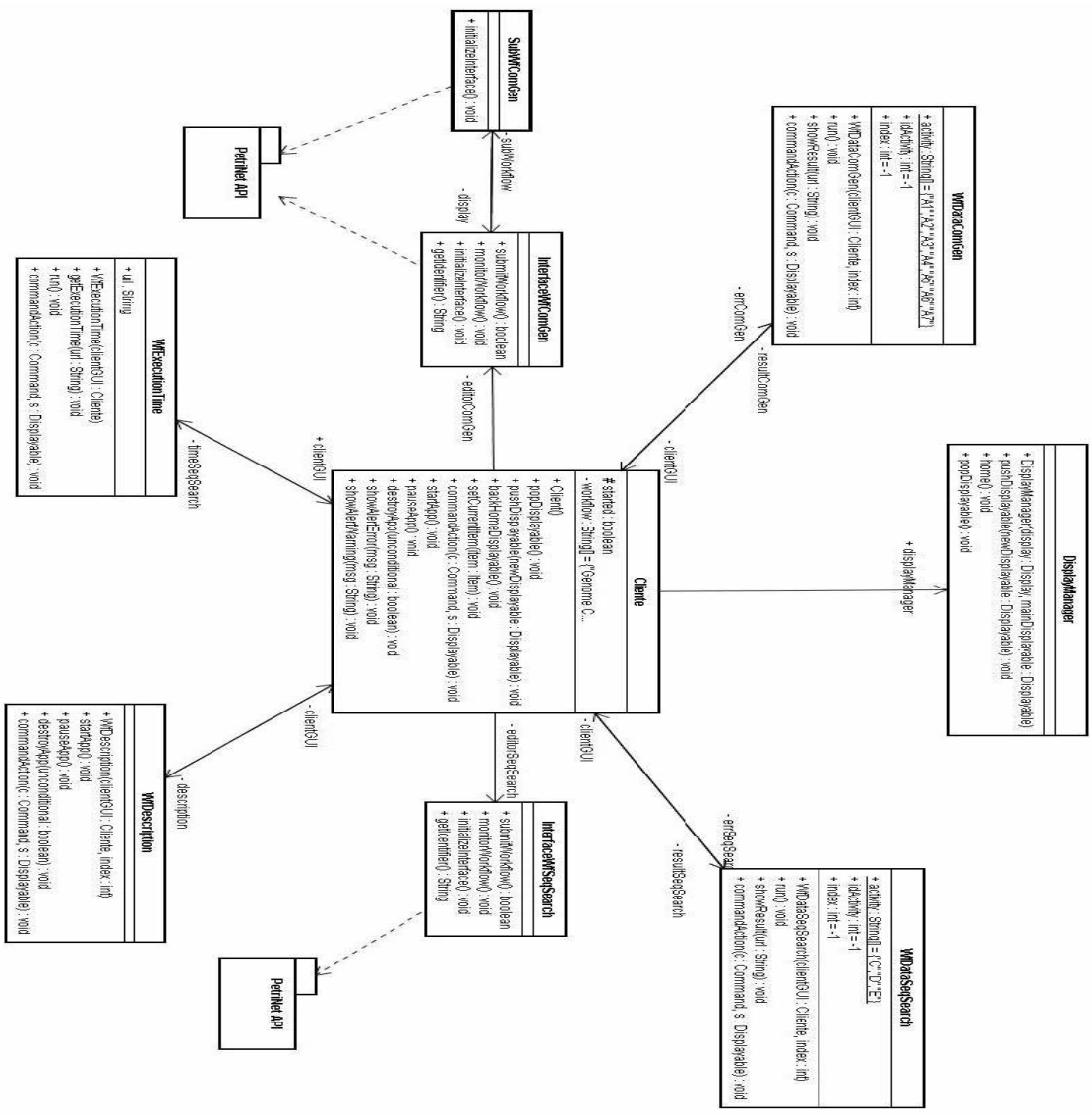

Figura 44: Diagrama de Classes do Portal

APÊNDICE D – Script Workflow

A figura 45 ilustra um exemplo de script *workflow* especificado em linguagem Karajan.

```
<project>
        <include file="sys.xml"/>
        <include file="task.xml"/>
     <set name="chromatdir" value="/mnt/win/phred/genome/chromat_dir/" ></set>
        <set name="teste">
                <host name="teste">
                        <service type= "execution" provider= "gt4" url= " https://
labweb02.inf.ufsc.br:8443/wsrf/services/ManagedJobFactoryService" jobManager="Fork"/>
                </host>
        </set>
       < execute executable= "/mnt/win/phred/genome/workflow/clean.exe" host= "{teste}"
provider="gt4" arguments="" stdout="/mnt/win/phred/genome/workflow/clean.out" stderr="/mnt/
win/phred/genome/workflow/clean.err"/>
       < execute executable= "/mnt/win/phred/genome/bin/phred" host="{teste}" provider= "gt4"
arguments= "{chromatdir} -pd /mnt/win/phred/genome/phd_dir/" stdout= "/mnt/win/phred/
genome/workflow/phred.out" stderr="/mnt/win/phred/genome/workflow/phred.err"/>
       < execute executable= "/mnt/win/phred/genome/bin/phd2fasta" host= "{teste}"
provider= "gt4" arguments= "-id /mnt/win/phred/genome/phd_dir/ -os /mnt/win/phred/genome/
workflow/seqs_fasta -oq /mnt/win/phred/genome/workflow/seqs_fasta.screen.qual" stdout= "/
mnt/win/phred/genome/workflow/phd2fasta.out" stderr= "/mnt/win/phred/genome/workflow/
phd2fasta.err"/>
       < execute executable= "/mnt/win/phred/genome/bin/phrap" host="{teste}" provider= "gt4"
arguments= "/mnt/win/phred/genome/workflow/seqs_fasta -minmatch 20 -new_ace" stdout= "/
mnt/win/phred/genome/workflow/phrap.out" stderr= "/mnt/win/phred/genome/workflow/
phrap.err"/>
       <parallel>
          < execute executable= "/mnt/win/glimmer2.13/run-glimmer2" host= "{teste}"
provider="gt4" arguments="/mnt/win/phred/genome/workflow/seqs_fasta.contigs" stdout="/mnt/
win/phred/genome/workflow/run-glimmer2.out" stderr="/mnt/win/phred/genome/workflow/run-
glimmer2.err"/>
          < execute executable= "/mnt/win/phred/genome/bin/tRNAscan-SE" host= "{teste}"
provider="gt4" arguments="/mnt/win/phred/genome/workflow/seqs_fasta.contigs" stdout="/mnt/
win/phred/genome/workflow/tRNAscan-SE.out" stderr= "/mnt/win/phred/genome/workflow/
tRNAscan-SE.err"/>
       </parallel>
       <parallel>
          < execute executable= "/mnt/win/phred/genome/bin/TransTerm" host= "{teste}"
provider="gt4" arguments="-s /mnt/win/phred/genome/workflow/seqs_fasta.contigs -c /mnt/win/
phred/genome/workflow/tmp.train -o /mnt/win/phred/genome/workflow/transterm.out" stdout="/
mnt/win/phred/genome/workflow/TransTerm.out" stderr= "/mnt/win/phred/genome/workflow/
TransTerm.err"/>
          < execute executable= "/mnt/win/phred/genome/bin/rbs_finder.pl" host= "{teste}"
provider= "gt4" arguments= "/mnt/win/phred/genome/workflow/seqs_fasta.contigs /mnt/win/
phred/genome/workflow/tmp.coord /mnt/win/phred/genome/workflow/rbs.out 50" stdout="/mnt/
win/phred/genome/workflow/rbs_finder.pl.out" stderr= "/mnt/win/phred/genome/workflow/
rbs_finder.pl.err"/>
          < execute executable= "/mnt/win/phred/genome/bin/hmmpfam" host= "{teste}"
provider= "gt4" arguments= "/mnt/win/phred/genome/databases/Pfam_fs /mnt/win/phred/
genome/workflow/tmp.train" stdout="/mnt/win/phred/genome/workflow/hmmpfam.out" stderr="/
mnt/win/phred/genome/workflow/hmmpfam.err" />
          < execute executable= "/mnt/win/blast/bin/blastall" host= "{teste}" provider= "gt4"
arguments="-p blastp -i /mnt/win/phred/genome/workflow/seqs_fasta.contigs" stdout="/mnt/win/
phred/genome/workflow/blastp.out" stderr="/mnt/win/phred/genome/workflow/blastp.err" />
       </parallel>
 </project>
```

Figura 45: Script Workflow em linguagem Karajan

APÊNDICE E – Descrição Semântica do Workflow

A figura 46 ilustra o arquivo em padrão xml de descrição semântica do primeiro exemplo de *workflow* implementado para validação da dissertação proposta. O arquivo mostra a funcionalidade, banco de dados e programa utilizados em cada tarefa.

```
<?xml version="1.0" encoding="UTF-8"?>
<descricao_workflow>
    <funcao_principal >O workflow tem a funcionalidade de pesquisar nucleotideos dentro de
alguns banco de dados de proteinas      em nodos de uma configuração grade. O pacote de
software Blast foi usado para pesquisar as sequências </funcao_principal>
      <tarefa nome="A">
            <funcao >É o começo da execução, responsável por inicializar o módulo Grid
Provider como sendo o Globus Toolkit 4.0.0</ funcao>
            <programa>NENHUM</programa>
            <bancodedados>NENHUM</bancodedados>
    </tarefa>
      <tarefa nome="B">
            <funcao>Deleta todos os arquivos gerados na execução anterior</ funcao>
            <programa>clean</programa>
            <bancodedados>NENHUM</bancodedados>
    </tarefa>
    <tarefa nome="C">
            <funcao >Invoca a aplicação Blast para pesquisar por uma específica sequencia
em uma banco de dados de proteinas </ funcao>
            <programa>Blastn</programa>
            <bancodedados>Ecoli</bancodedados>
    </tarefa>
      <tarefa nome="D">
            <funcao >Invoca a aplicação Blast para pesquisar por uma específica sequencia
em uma banco de dados de proteinas</ funcao>
            <programa>Blastn</programa>
            <bancodedados>MITO</bancodedados>
    </tarefa>
    <tarefa nome="E">
            <funcao >Invoca a aplicação Blast para pesquisar por uma específica sequencia
em uma banco de dados de proteinas</ funcao>
            <programa>Blastn</programa>
            <bancodedados>IGSEQPROT</ bancodedados>
    </tarefa>
      <tarefa nome="F">
            <funcao >transfere todos os arquivos (que resultam das atividades C,D e E) para
uma localização onde o usuário móvel pode acessar</ funcao>
            <programa>cp</programa>
            <bancodedados>NENHUM</bancodedados>
    </tarefa>
</descricao_workflow>
```

Figura 46: Descrição do Primeiro Exemplo Workflow

A figura 47 ilustra o arquivo em padrão xml de descrição semântica do segundo exemplo de *workflow* implementado para validação da dissertação proposta.

```
<?xml version="1.0" encoding="UTF-8"?>
<descricao_workflow>
     < funcao_principal   >Workflow para analisar e anotar a bactéria Gluconacetobacter
diazotrophicus.</funcao_principal>
        <tarefa nome="A1">
                <funcao >O primeiro passo é gerar os reads e contigs dos cromatogramas da
bactéria, obtidos nos laboratórios.</funcao>
                <programa>Phrep, Phd2fasta, Phrap</programa>
                <bancodedados>NENHUM</bancodedados>
     </tarefa>
        <tarefa nome="A2">
                <funcao>Ele trata da predição de genes. Como Glimmer pode gerar falso positivos
e falso negativos, pesquisadores analisam outras informações para identificar um gene, como
por exemplo a presência de tRNAs (um outro tipo gene não detectado pelo Glimmer),
terminador (tarefa A4) e sites de binding ribossome (tarefa A5). Na presente tarefa ocorre a
busca por tRNAs.</funcao>
                <programa>tRNAscan-SE</programa>
                <bancodedados>NENHUM</bancodedados>
     </tarefa>
     <tarefa nome="A3">
                <funcao >Ele obtém ORFs (open reading frames), que são possíveis genes</
funcao>
                <programa>Glimmer</programa>
                <bancodedados>NENHUM</bancodedados>
     </tarefa>
        <tarefa nome="A4">
                <funcao>Procura terminador</funcao>
                <programa>Transterm</programa>
                <bancodedados>NENHUM</bancodedados>
     </tarefa>
     <tarefa nome="A5">
                <funcao>Analisa sites de binding ribosome</ funcao>
                <programa>rbs_finder.pl</programa>
                <bancodedados>NENHUM</bancodedados>
     </tarefa>
        <tarefa nome="A6">
                <funcao >Trata do descobrimento da função do gene, que é feita através da
comparação da sequência de ORF com outras sequências já existentes em dados públicos.
Usando um programa para coleção de família de proteínas e domínios armazenados em um
banco de dados</funcao>
                <programa>HMMPFam</programa>
                <bancodedados>PFam</bancodedados>
     </tarefa>
     <tarefa nome="A7">
                <funcao >Trata do descobrimento da função do gene, que é feita através da
comparação da sequência de ORF com outras sequências já existentes em dados públicos.
Usando um programa para coleção de família de proteínas e domínios armazenados em um
banco de dados</funcao>
                <program>Blastp</program>
                <bancodedados>NR</bancodedados>
     </tarefa>
</descricao_workflow>
```

Figura 47: Descrição do Segundo Exemplo Workflow

APÊNDICE F – Questinário de Avaliação

A figura 48 ilustra o questionário de avaliação elaborado por um especialista em usabilidade para melhorar a qualidade das interfaces de forma geral.

Questionário de Avaliação

Em uma escala que vai de 1(um) a 5(cinco), informe a sua avaliação do sistema. O 1(um) corresponde a uma avaliação baixa e o 5 (cinco) equivale a uma avaliação alta.

	1	2	3	4	5
Você possui bons conhecimentos de informática?					X
Possui familiaridade com dispositivos móveis?			X		
Qual avaliação você faz da facilidade de utilizar o sistema?				X	
Em relação ao nível de aprendizado, como você o avalia?					X
Você ficou satisfeito com a interface do sistema?			X		
Você gostaria de utilizar este sistema?			X		
O sistema desempenha o papel que você gostaria?					X
Você achou o sistema confortável de utilizar?		X			
Você possui o conhecimento especializado para utilizar o sistema?					X

Figura 48: Questionário de Avaliação

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo