UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

ADENAUER CORRÊA YAMIN

Arquitetura para um Ambiente de Grade Computacional Direcionado às Aplicações Distribuídas, Móveis e Conscientes do Contexto da Computação Pervasiva

Tese apresentada como requisito parcial
para obtenção do grau de
Doutor em Ciência da Computação

Prof. Dr. Cláudio Fernando Resin Geyer
Orientador

Porto Alegre, agosto de 2004

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Yamim, Adenauer Corrêa

Arquitetura para um Ambiente de Grade Computacional Direcionado às Aplicações Distribuídas, Móveis e Conscientes do Contexto da Computação Pervasiva / Adenauer Corrêa Yamin – Porto Alegre: Programa de Pós-Graduação em Computação, 2004.

195 f. : il.

Tese (doutorado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação. Porto Alegre, BR-RS, 2004. Orientador: Cláudio Fernando Resin Geyer.

1. Computação Pervasiva. 2. Computação em Grade. 3. Computação Móvel. 4. Arquitetura de Software. 5. *Middleware* Adaptativo. Aplicações Conscientes do Contexto.I. Geyer, Cláudio Fernando Resin. II. Título.

À minha esposa Paula, e filhas Amanda e Lara
pela aceitação de minhas ausências.

AGRADECIMENTOS

Pesquisar é uma atividade dinâmica que prospera em um ambiente que estimule a troca de idéias e uma intensa colaboração entre os envolvidos. Encontrei este ambiente no Grupo de Pesquisa em Processamento Paralelo e Distribuído - GPPD - do Instituto de Informática da UFRGS. Foi a interação com os investigadores entusiasmados e cooperativos deste grupo que conduziu aos resultados atingidos neste documento. Dizer que sem eles este trabalho não teria sido possível é uma visão parcial. Este trabalho é deles, tanto quanto meu. Um ambiente de pesquisa assim, é o contra-ponto para a noção estereotipada do pesquisador solitário, que se instala em uma torre e dela volta com uma tese. Com alegria apresento neste documento os resultados de um esforço verdadeiramente colaborativo. Neste esforço compartilhei aprendizados não somente sobre ciência da computação, mas também como melhor posicionar minhas idéias, defendê-las e escrever sobre as mesmas.

A intencionalidade de cada vez mais apreciar o convívio da dúvida, e recusar sistematicamente o conforto do conhecido, se impõe como um desafio extenuante durante um curso de doutorado. Muitos foram fundamentais no dia-a-dia desta caminhada, em particular quero agradecer:

à amiga Iara, pelo companheirismo e dedicação que sempre dispensou no dia-a-dia da pesquisa que propusemos desvendar junto.

aos CWs Luciano e Rodrigo, parceiros nos embates, pela estimulante troca de idéias e apoio irrestrito.

aos colegas Alberto e Gustavo, por suas contribuições em diversas implementações deste trabalho.

ao amigo Jorge, pelo constante estímulo e encorajamento.

ao meu orientador, Cláudio Geyer, pelo suporte e amizade oferecidos ao longo dos anos que temos trabalhado juntos.

à UCPel, UFPel e CAPES, pelo apoio institucional ao trabalho, a FAPERGS pelo apoio financeiro ao Projeto ISAM, e ao CNPq/FINEP/SEPIN os recursos concedidos ao Projeto contextS.

ao Instituto de Informática da UFRGS, na figura de seus professores e funcionários, pelas inúmeras lições de como ensinar, pesquisar e promover ciência com dedicação e ética.

Por fim, registro meu sempre presente agradecimento ao Grande Arquiteto do Universo a oportunidade desta excepcional experiência de vida.

Ventana sobre la utopía

Ella está en el horizonte -dice Fernando Birri-. Me acerco dos pasos, ella se aleja dos pasos. Camino diez pasos y el horizonte se corre diez pasos más allá. Por mucho que yo camine, nunca la alcanzaré. ¿Para que sirve la utopía? Para eso sirve: para caminar.

Eduardo Galeano, Las Palabras Andantes, 1993

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
APPELO	Ambiente de Programação Paralela em Lógica
AVU	Ambiente Virtual do Usuário
BDA	Base de Dados pervasiva das Aplicações
CIB	*Cell Information Base*
CORBA	*Common Object Request Broker Architecture*
DC	*Dinamic Configurator*
EXEHDA	*Execution Environment for Highly Distributed Applications*
EXEHDA-AMI	*EXEHDA Architecture Managenment Interface*
GSM	*Global System for Móbiles*
HOLO	Holoparadigma
IETF	*Internet Engineering Task Force*
ISAM	Infra-estrutura de Suporte às Aplicações Móveis Distribuídas
ISAMpe	*ISAM Pervasive Environment*
J2ME	*Java 2 Platform, Micro Edition*
J2SE	*Java 2 Platform, Standard Edition*
JVM	*Java Virtual Machine*
LAN	*Local Área Network*
LDAP	*Lightweight Directory Access Protocol*
OPERA	*Or Parallel Prolog*
OX	Objeto eXehda
P2P	*Peer-to-Peer Computing*
PDA	*Personal Digital Assistent*
PRIMOS	*PRIMitives for Object Scheduling*
PvC	*Pervasive Computing*
RMI	*Remote Method Interface*
RNP	Rede Nacional de Pesquisas
RPC	*Remote Procedure Call*
SNMP	*Simple Network Management Protocol*
TCP/IP	*Trasmission Control Protocol/Internet Protocol*
UML	*Unified Modeling Language*
XML	*Extended Modeling Language*

LISTA DE FIGURAS

LISTA DE QUADROS

RESUMO

Neste início de década, observa-se a transformação das áreas de Computação em Grade (*Grid Computing*) e Computação Móvel (*Mobile Computing*) de uma conotação de interesse emergente para outra caracterizada por uma demanda real e qualificada de produtos, serviços e pesquisas. Esta tese tem como pressuposto a identificação de que os problemas hoje abordados isoladamente nas pesquisas relativas às computações em grade, consciente do contexto e móvel, estão presentes quando da disponibilização de uma infra-estrutura de software para o cenário da Computação Pervasiva. Neste sentido, como aspecto central da sua contribuição, propõe uma solução integrada para suporte à Computação Pervasiva, implementada na forma de um *middleware* que visa criar e gerenciar um ambiente *pervasivo*, bem como promover a execução, sob este ambiente, das aplicações que expressam a semântica *siga-me*. Estas aplicações são, por natureza, distribuídas, móveis e adaptativas ao contexto em que seu processamento ocorre, estando disponíveis a partir de qualquer lugar, todo o tempo. O *middleware* proposto, denominado EXEHDA (*Execution Environment for Highly Distributed Applications*), é adaptativo ao contexto e baseado em serviços, sendo chamado de ISAMpe o ambiente por este disponibilizado. O EXEHDA faz parte dos esforços de pesquisa do Projeto ISAM (Infra-Estrutura de Suporte às Aplicações Móveis Distribuídas), em andamento na UFRGS. Para atender a elevada flutuação na disponibilidade dos recursos, inerente à Computação Pervasiva, o EXEHDA é estruturado em um núcleo mínimo e em serviços carregados sob demanda. Os principais serviços fornecidos estão organizados em subsistemas que gerenciam: (*a*) a execução distribuída; (*b*) a comunicação; (*c*) o reconhecimento do contexto; (*d*) a adaptação; (*e*) o acesso *pervasivo* aos recursos e serviços; (*f*) a descoberta e (*g*) o gerenciamento de recursos. No EXEHDA, as condições de contexto são pró-ativamente monitoradas e o suporte à execução deve permitir que tanto a aplicação como ele próprio utilizem essas informações na gerência da adaptação de seus aspectos funcionais e não-funcionais. O mecanismo de adaptação proposto para o EXEHDA emprega uma estratégia colaborativa entre aplicação e ambiente de execução, através da qual é facultado ao programador individualizar políticas de adaptação para reger o comportamento de cada um dos componentes que constituem o software da aplicação. Aplicações tanto do domínio da Computação em Grade, quanto da Computação Pervasiva podem ser programadas e executadas sob gerenciamento do *middleware* proposto.

Palavras-Chave: Computação Pervasiva. Computação em Grade. Computação Móvel. Arquitetura de Software. *Middleware* Adaptativo. Aplicações Conscientes do Contexto.

Architecture for a Computational Grid Environment Addressed to Distributed, Mobile and Context Aware Applications in Pervasive Computing

ABSTRACT

Transformations have been witnessed in the areas of Grid Computing and Mobile Computing at the beginning of this decade. They go from having an emergent interest status to being characterized by a real and qualified demand of products, services and research. This PhD thesis is based on the belief that the problems that are addressed separately today in research related to grid, context-sensitive and mobile computing are also present in the creation of a software infra-structure geared towards the pervasive computing scenario. The central aspect of the thesis contribution is the proposal of an integrated solution for the support of pervasive computing, implemented as a middleware which aims at creating and managing a pervasive environment, as well as allowing the execution, in this environment, of follow-me applications. These applications are, by nature, distributed, mobile and capable of adapting to the context in which their processing occurs, being available anywhere, anytime. The proposed middleware, named EXEHDA (Execution Environment for Highly Distributed Applications), is context adaptable and service based, and the environment made available by this middleware is called ISAMpe. EXEHDA is part of the research efforts of the ISAM project (Infra-Estrutura de Suporte às Aplicações Móveis Distribuídas), which is ongoing (sugestão: under development) at UFRGS. To deal with the big variance in the availability of resources, which is inherent to pervasive computing, EXEHDA is structured as a minimum core and as services loaded on demand. The main services supplied are organized in subsystems that manage: (*a*) distributed execution; (*b*) communications; (*c*) context recognition; (*d*) adaptation; (*e*) pervasive access to resources and services; (*f*) discovery and (*g*) management of resources. In EXEHDA, the context conditions are actively monitored, and the execution support must allow for both the application and the middleware to use this information in the management of the adaptation of their functional and non-functional aspects. The adaptation mechanism proposed for EXEHDA employs a collaborative strategy between the application and the execution environment, through which the programmer may individualize the adaptation policies that will control the behavior of each of the components that constitute the application software. Applications from the grid computing domain as well as applications from the pervasive computing domain may be programmed and executed under the management of the proposed middleware.

Keywords: Pervasive Computing. Grid Computing. Mobile Computing. Software Architecture. Adaptive Middleware. Context-Aware Applications.

1 INTRODUÇÃO

A problem well stated is a problem half solved.

- Charles Kettering

Neste início de década, é observado um movimento em direção à Computação Pervasiva [GRI 2001, GRI 2001a], que contempla aplicações com novas funcionalidades. Computação Pervasiva é a proposta de um novo paradigma computacional, que permite ao usuário o acesso a seu ambiente computacional a partir de qualquer lugar, todo o tempo [SAH 2003]. O usuário poderá utilizar equipamentos com diferentes perfis de hardware, que poderão ter suporte a operação móvel ou não. O surgimento de publicações em editoriais internacionalmente reconhecidos atesta a acelerada disseminação desta proposta [IPC 2002, TMC 2002].

O emprego do termo Computação Pervasiva ficou associado à IBM quando da edição entitulada *Pervasive Computing* do IBM System Journal [IBM 99], onde foi organizada uma digressão sobre os aspectos promissores da Computação Pervasiva, tendo sido também, nesta mesma edição, resgatada por Weiser, no artigo entitulado "*The origins of ubiquitous computing research at PARC in the late 1980s*", a sua visionária proposta quanto ao futuro da computação, na qual recursos de computação onipresentes se ajustariam, de forma autônoma, para atender os usuários.

Embora a proposta original de Weiser quanto à "Computação Ubíqua" ainda esteja distante de uma prática cotidiana alicerçada por produtos de mercado [SAT 2001], sua proposta vem se materializando, pouco a pouco, através da disponibilização de tecnologias como PDAs, SmartPhones e a consolidação de padrões para redes sem fio como o Bluetooth [BLU 2002] e o IEEE 802.11 [WIF 2002].

A Computação Pervasiva, plataforma escopo desta tese, se mostra uma etapa indispensável a ser consolidada no caminho de propostas como a de Weiser. Mesmo tendo uma premissa mais próxima das tecnologias de hardware e software atualmente praticadas, a Computação Pervasiva constituirá ainda um campo fértil para ofertas de produtos e desenvolvimento de pesquisas nos próximos anos [SAT 2001].

Este novo cenário prevê uma mobilidade física (dos equipamentos e/ou dos usuários) e do software (componentes da aplicação e serviços). Para isto, o sistema de suporte à execução deve permitir que o usuário possa acessar seu ambiente computacional independente de localização e de tempo. Este sistema usa a metáfora de um ambiente virtual do usuário, onde as aplicações têm o estilo *siga-me* (*follow-me applications*). Logo, para viabilizar esse novo estilo de programação é necessário uma infra-estrutura de suporte para o projeto, implementação e execução do software a ser desenvolvido. Este é o objetivo do projeto ISAM - Infra-estrutura de Suporte às Aplicações Móveis Distribuídas -, em desenvolvimento no II/UFRGS desde 2001 [ISA 2002, AUG 2002b, YAM 2002].

A tese defendida é que a infra-estrutura de suporte à *pervasividade* em escala global pode ser construída através da integração de três áreas da computação: Computação Móvel, Computação em Grade e Computação Consciente do Contexto [YAM 2003]. Esta visão é diferente da de outros projetos de infra-estrutura para a Computação Pervasiva, tais como Aura [GAR 2002] e Gaia [ROM 2002], que propõem soluções focadas em uma visão de computação pessoal e em uma visão de contexto local, respectivamente.

O trabalho relatado neste texto está inserido na agenda de pesquisas do projeto ISAM como um todo. Particularmente, são aprofundados os aspectos relativos ao subprojeto EXEHDA (*Execution Environment for Highly Distributed Applications*) [YAM 2003a, YAM 2004, EXE 2003], o qual propõe soluções para o ambiente de suporte à programação e à execução de aplicações ISAM, implementado na forma de um *middleware*.

Este capítulo tem por objetivo contextualizar a tese proposta, bem como apresentar suas motivações e objetivos. Constituído de 6 seções, ele também contempla aspectos da organização do texto como um todo.

1.1 Tema

Esta tese tem por foco definir a arquitetura para um ambiente de execução destinado às aplicações da Computação Pervasiva, implementado na forma de um *middleware* denominado EXEHDA [YAM 2003, YAM 2003a, YAM 2004]. No EXEHDA, as condições de contexto são pró-ativamente monitoradas, e o suporte à execução deve permitir que tanto a aplicação como ele próprio utilizem estas informações na gerência da adaptação de seus aspectos funcionais e não-funcionais. Também a premissa *siga-me* das aplicações *pervasivas* deverá ser suportada, garantindo a execução da aplicação do usuário em qualquer tempo e lugar [YAM 2001, AUG 2001].

As aplicações-alvo são distribuídas, adaptativas ao contexto em que executam e compreendem a mobilidade lógica e a física, sendo baseadas no modelo de programação adotado no projeto ISAM, o ISAMadapt [AUG 2004]. O ISAMadapt partiu das abstrações do Holoparadigma [BAR 2002a] e adicionou novas abstrações e construções com uma semântica adequada ao ambiente *pervasivo*. Na perspectiva do ISAM, entende-se por mobilidade lógica a movimentação entre equipamentos de artefatos de software e seu contexto [FUG 98], e por mobilidade física o deslocamento do usuário, portando ou não seu equipamento [ISA 2002].

O mecanismo de adaptação ao contexto do EXEHDA propõe uma estratégia colaborativa entre aplicação e ambiente de execução, através da qual é facultado ao programador individualizar políticas de adaptação para reger o comportamento de qualquer componente da aplicação [YAM 2002b]. As políticas que irão reger os mecanismos de adaptação, funcionais ou não, são especificadas pelo ambiente de desenvolvimento provido pelo ISAMadapt [AUG 2004].

Na concepção e desenvolvimento do EXEHDA foram considerados temas das seguintes áreas da ciência da computação: programação em sistemas distribuídos e paralelos [YAM 99], gerenciamento de recursos [YAM 2000, YAM 2001a, NOG 2001], arquitetura de software [GRI 95, SHA 96], adaptação em sistemas computacionais [YAM 2002a, YAM 2002b], consciência do contexto [AUG 2003, YAM 2003, CHE 2001] e Computação Móvel e em grade [GEY 2002, YAM 2003, YAM 2002, YAM 2001, AUG 2001].

1.2 Motivação

Previsões indicam que os próximos anos serão caracterizados por elevados níveis de heterogeneidade e pela interação entre dispositivos conectados a redes de abrangência global [SAH 2003, SAT 2001]. Estas redes interligadas utilizarão tanto conexões cabeadas como sem fio. As pesquisas envolvendo sistemas distribuídos em redes de grande abrangência (*wide-area*) responderam a diversas questões pertinentes ao acesso aos recursos neste tipo de ambiente; contudo, existem lacunas no que diz respeito ao tratamento dinâmico da adaptação à medida que as aplicações são executadas [REA 2003a, AUG 2002a, BUY 2001].

Tanto a disseminação física da Internet como o aumento da velocidade operacional das redes de computadores que a compõem conduz a uma perspectiva de uso unificado dos recursos distribuídos, o qual pode ser realizado a partir de equipamentos posicionados em qualquer local das redes interconectadas. Esta situação vem materializando as premissas da Computação em Grade (*Grid Computing*) [BUY 2001, FOS 2001].

De forma similar, o crescente uso das redes sem fio vem conduzindo a uma proposta efetiva de Computação Móvel (*Mobile Computing*) [AUG 2001, CHE 2001], na qual o usuário utilizando dispositivos portáteis, como *palmtops* e *notebooks*, poderá ter acesso a uma infra-estrutura de serviços e manterá este acesso durante seu deslocamento.

Considerando o início das atividades de pesquisa correspondentes ao EXEHDA, registrada com a publicação [YAM 2001], foi possível observar a transformação das áreas de Computação Móvel e Computação em Grade, de uma conotação de interesse emergente, para outra caracterizada por uma demanda real e qualificada de produtos, serviços e pesquisas.

Em contraste com a premissa clássica da computação distribuída [BAL 89] que busca liberar os programadores de considerar o estado do meio físico aonde acontece o processamento, esta nova classe de aplicações é sensível ao contexto onde ocorre a execução (*context-aware* [CHE 2001, CTX 2003]), e o programador deve prever o uso destas informações na gerência dos seus aspectos funcionais e não-funcionais. As condições de contexto são pró-ativamente monitoradas e o modelo de execução deve permitir que tanto a aplicação como ele próprio reajam às alterações no ambiente através de mecanismos de adaptação. Este processo requer a existência de múltiplos caminhos de execução, ou de configurações alternativas, as quais exibem diferentes perfis de utilização dos recursos computacionais. A figura 1.1 caracteriza, sob a óptica do autor, o amadurecimento em direção à Computação Pervasiva. Esta visão foi fortemente influenciada pelos trabalhos desenvolvidos no âmbito do Projeto ISAM.

A efetividade de um comportamento adaptativo ao contexto durante a execução de uma aplicação distribuída, em ambiente com suporte à mobilidade física, tanto do usuário como do equipamento, implica a resposta a diversas questões. Considerando o foco de pesquisa do EXEHDA, destaca-se o problema de “como disponibilizar um ambiente computacional pervasivo e gerenciar suas aplicações?”. Deste problema derivam outras questões:

- como suportar a típica condição da Computação Pervasiva de elevada heterogeneidade na efetiva disponibilidade dos recursos, isto mesmo quando os equipamentos envolvidos são constituídos de hardware e software similares?

- como garantir, na perspectiva da mobilidade lógica e física, a relação entre os componentes de hardware e software ativos da aplicação?
- como garantir a disponibilidade de dados e código na perspectiva da mobilidade, ou seja, a partir de qualquer posição da infra-estrutura de rede?
- como preservar a consistência da execução distribuída, na presença de desconexões assíncronas de nodos de processamento?
- como gerenciar, de forma personalizada, a disponibilização do estado atual do contexto com significado para cada aplicação?
- como deve ser a arquitetura do ambiente de execução para que os processos de adaptação ao contexto ocorram de forma integrada aos mecanismos de gerência do processamento?
- como perseguir interoperabilidade, considerando a pluralidade de natureza do hardware e sistemas operacionais na Computação Pervasiva?
- Estas questões ressaltam que, independentemente dos mecanismos de consciência do contexto e de gerência da adaptação, a premissa *siga-me* da Computação Pervasiva deverá ser suportada, garantindo a execução da aplicação do usuário a partir de qualquer localização e tipo de equipamento, todo o tempo [YAM 2002].

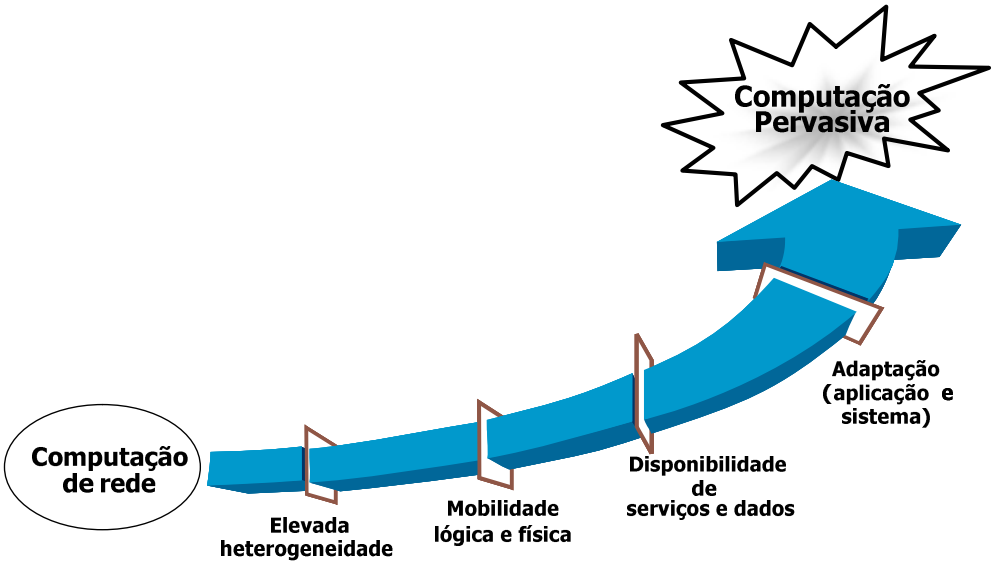


Figura 1.1: Consolidação do cenário da Computação Pervasiva

Do ponto de vista tecnológico, um computador portátil deve ser pequeno, leve, e requerer fontes reduzidas de energia. Isto significa que este tipo de equipamento apresenta restrições no tamanho da memória, na capacidade de armazenamento, no consumo de energia por parte do processador, bem como na interface disponibilizada ao usuário.

Os avanços na microeletrônica nos últimos anos têm viabilizado equipamentos com um consumo cada vez menor de energia, aumentando, deste modo, a oferta de produtos

que potencializam a mobilidade (PDAs, adaptadores *wireless*, etc.). Esta leitura é visível pela crescente qualidade tanto do hardware como do software disponibilizado nos sítios especializados [GPB 2003, GEA 2003]. Também, como indicador do crescimento da área de Computação Pervasiva e dos recursos para suportá-la, tem-se o surgimento de revistas com editoriais internacionalmente reconhecidos [IPC 2002, TMC 2002], e a gradativa inserção de temas relacionados à mesma em congressos de praticamente todas as áreas da computação.

Por outro lado, os avanços na microeletrônica também têm promovido a oferta de processadores e padrões de rede para *desktops* cada vez mais poderosos.

A superação da elevada disparidade nas características dos recursos que compõem o sistema distribuído é um desafio cada vez mais presente na proposta da Computação Pervasiva. Nesta direção surge o primeiro foco motivacional do EXEHDA.

1.2.1 Primeiro foco motivacional – dinamicidade e heterogeneidade do ambiente de processamento

O deslocamento do usuário (mobilidade física), aspecto característico da Computação Pervasiva, determina a execução de aplicações a partir de diferentes equipamentos e/ou pontos da rede global, nos quais a oferta e/ou disponibilidade dos recursos computacionais é variável. Disto decorre:

- **elevada flutuação na banda passante disponível para as comunicações**: a diversidade de tecnologias envolvidas provoca flutuações da ordem de Kbytes até Mbytes na banda-passante praticada nas comunicações. Esta situação fica potencializada nas redes sem fio, cujos comportamentos operacionais estão sujeitos a um espectro ainda maior de influências. Um exemplo de adaptação oportuna, neste caso, é a troca nas especificações dos dados transmitidos (compactação, redução de cor e/ou resolução, etc.), mantendo o comportamento semântico da aplicação;

- **equipamentos de usuários com acentuadas diferenças nos atributos de hardware e sistema operacional**: mesmo no contexto de Computação em Grade em que os equipamentos são fixos, porém a gerência é multi-institucional (portanto, descentralizada e de responsabilidade de cada localidade), os diferentes posicionamentos dos usuários/dispositivos (mobilidade física) levam a um uso de recursos computacionais diversificados. Por sua vez, considerando o emprego da mobilidade lógica, uma aplicação poderá, em alguns casos, ser manipulada tanto por equipamentos de médio/grande (*desktops*), como de pequeno porte (PDAs). Estes equipamentos apresentam significativas diferenças nas configurações de processador, memória, periféricos, etc., ficando ainda mais potencializada a heterogeneidade a que será exposta a aplicação final. Estratégias envolvendo o mapeamento de recursos, balanceamento de carga, e a seleção dinâmica do código a ser utilizado para tratamento de entrada/saída, são exemplos de adaptações neste cenário;

- **diferentes infra-estruturas para conexão à rede global**: no ambiente móvel conforme o usuário se desloca, a célula que suporta

> a conexão do dispositivo móvel se modifica e, conseqüentemente, se alteram aspectos como: serviços localmente disponíveis, o hardware para suporte aos mesmos, etc. O ambiente de execução, sempre que necessário, deve prover uma visão global dos recursos que oferece e, sob demanda, carregar os módulos do *middleware* adequados às necessidades da aplicação.

Por outro lado, a perspectiva da Computação Pervasiva contempla a coexistência de aplicações distribuídas de diferentes naturezas: áudio, vídeo, textos, cálculos numericamente intensivos, etc. Deste fato decorre a segunda motivação do EXEHDA.

1.2.2 Segundo foco motivacional – não comprometimento com uma classe específica de aplicações adaptativas

A idéia de sistemas adaptativos não é nova. Segundo Katz [KAT 94], mobilidade implica adaptabilidade, o que significa que sistemas devem ter consciência da localização e do contexto, e devem tirar vantagem desta informação, estruturando-se de modo distribuído e reconfigurando-se dinamicamente. Modelos, arquiteturas e tecnologias para a Computação Móvel estão, no entanto, ainda nos seus primeiros estágios de desenvolvimento, e o enfrentamento dos desafios postos pela mobilidade está iniciando [ROM 2000].

Sensível a esta necessidade, a comunidade científica vem enfrentando estes desafios [CHE 2001, GRI 2001], tendo surgido alguns trabalhos contemplando a gerência de aplicações com suporte à mobilidade lógica e/ou física (software e/ou hardware). Dentre estes, citam-se [LUN 2001, NOB 2000, KUN 99, BAG 98, RAN 97].

Estes trabalhos apresentam aspectos comuns: (*i*) atuam gerenciando as larguras de banda utilizadas nas comunicações, (*ii*) contemplam um domínio específico para as aplicações, tais como: multimídia e informações dependentes do contexto e, (*iii*) usualmente implementam somente uma estratégia de adaptação: alteração no formato dos dados, replicação de dados ou migração de tarefas.

Parece, portanto, ser necessário definir uma nova arquitetura de sistemas móveis, projetada desde seu início com flexibilidade e adaptabilidade em mente.

Perseguindo esta visão, o EXEHDA integra os esforços de pesquisa do Projeto ISAM [ISA 2002], contribuindo na proposição de uma arquitetura de software que simplifique a tarefa de implementação e execução de aplicações *pervasivas*. De uma forma resumida, o objetivo é conceber um ambiente de execução no qual todos os componentes estão comprometidos com as propriedades de “pervasividade” e adaptabilidade ao contexto. A perspectiva deste comprometimento é suportar a execução de aplicações em diferentes contextos e conseqüentes necessidades de adaptação. Desta propriedade surge a terceira motivação do EXEHDA.

1.2.3 Terceiro foco motivacional – controle da adaptação

A gerência da adaptação pode ocorrer no limite de dois extremos: (*i*) no primeiro, denominado *laissez-faire*, a aplicação é responsável por toda a adaptação que será realizada; por sua vez, no segundo extremo, (*ii*) denominado *application-transparent*, o sistema é encarregado de gerenciar toda a adaptação que vier a ocorrer. Dois trabalhos são marcos de referência para estes dois limites, [CEN 97] e [SAT 90], respectivamente.

Na perspectiva *application-transparent*, o *middleware* oculta da aplicação toda complexidade inerente à gerência da adaptação. Deve tratar de forma transparente com

diferentes tipos de problemas e de forma autônoma deve procurar a melhor configuração para a aplicação em execução. Estudos feitos e registrados nas referências [YAM 2000], [YAM 2001a] e [YAM 2001] indicam que um *middleware* com este perfil implica uma estrutura de software com custo computacional elevado. Este custo diz respeito tanto ao montante de código envolvido, como à complexidade algorítmica e ao volume de dados que precisam ser monitorados e manipulados pelos mecanismos de controle da adaptação.

É importante ressaltar que um *middleware* direcionado para um meio físico de execução heterogêneo não pode ter um custo computacional elevado como premissa de projeto. Este aspecto se potencializa no caso do EXEHDA, pois é contemplado o suporte aos *palmtops* atualmente empregados na Computação Móvel, cujo porte contrasta significativamente com aquele dos *desktops* empregados na Computação Distribuída e em Grade.

Por sua vez, na proposta *laissez-faire* de adaptação, a ausência de mecanismos que realizem algum tipo de centralização inviabiliza uma contemporização de interesse das diferentes partes da aplicação distribuída em execução. Soma-se a isto, o fato de que o desenvolvimento de aplicações neste modelo é trabalhoso devido, sobretudo, à elevada personalização e ao conseqüente baixo reaproveitamento do código já existente [YAM 2000].

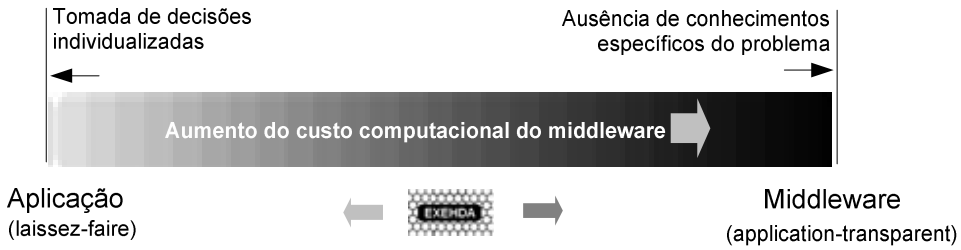


Figura 1.2: Responsabilidade da decisão de adaptação

A priori, nenhuma dessas estratégias pode ser considerada a melhor. Uma estratégia diferente pode ser requerida para cada circunstância e/ou aplicação. Há situações, por exemplo, onde o código fonte da aplicação não está disponível e a estratégia a ser utilizada deve ser a *application-transparent*. Em outros casos, pode ser mais oportuno incluir apenas na aplicação os mecanismos adaptativos, sem envolver o ambiente de execução.

Logo, a proposta para o EXEHDA é modelar um *middleware* que faculte uma estratégia colaborativa com a aplicação nos procedimentos de adaptação (figura 1.2). Deste modo, considerando a natureza da aplicação, o programador poderá definir a distribuição de responsabilidades entre o *middleware* e a aplicação no processo de adaptação [YAM 2002b, AUG 2002b].

Por fim, da propriedade de mobilidade da Computação Pervasiva advém a quarta motivação.

1.2.4 Quarto foco motivacional – suporte às mobilidades lógica e física

A Computação Móvel genericamente se refere a um cenário onde todos, ou alguns nodos que tomam parte no processamento, são móveis [BAG 98, AUG 2001]. Desta definição podem derivar diferentes interpretações. Em um extremo, a mobilidade leva

em conta as necessidades dos usuários nômades, isto é, usuários cuja conexão na rede ocorre de posições arbitrárias e que não ficam permanentemente conectados. Em outro extremo, estão os usuários móveis, os quais retêm a conectividade durante o deslocamento, tipicamente explorando conexões sem fio. Desta forma, a Computação Móvel é caracterizada por três propriedades: mobilidade, portabilidade e conectividade [ROM 2000]. Na proposta do EXEHDA estas propriedades desdobram duas preocupações de pesquisa:

- os segmentos sem fio da rede global levantam novas condições operacionais, entre as quais se destaca a comunicação intermitente. A ocorrência de desconexões de *nodos* no ambiente móvel, sejam estas voluntárias ou não, é mais uma regra do que uma exceção;
- a natureza dinâmica do deslocamento do hardware e do software na rede global introduz questões relativas tanto à identificação física dos nodos quanto à localização dos componentes de software que migram. Estas questões apontam para a necessidade de mecanismos dinâmicos que realizem o mapeamento dos componentes móveis, de modo a viabilizar sua localização e permitir a interação com os mesmos.

Portanto, o *middleware* para suporte à Computação Pervasiva deve levar em conta essas limitações, de modo que as aplicações não percam sua consistência quando um componente de software migrar entre nodos da rede global, ou quando um nodo temporariamente não estiver disponível por estar desligado ou sem conexão, ou ainda trocar sua célula de execução em função do deslocamento. Enfim, o *middleware* deverá viabilizar a semântica *siga-me* das aplicações *pervasivas*.

A localização é um aspecto-chave para os sistemas com mobilidade, pois a localidade influi significativamente no contexto disponibilizado para os mecanismos de adaptação. O contexto representa uma abstração peculiar da Computação Pervasiva, e inclui informações sobre recursos, serviços e outros componentes do meio físico de execução. Nesta ótica, o ambiente de execução deve fornecer informações de contexto que extrapolam a localização onde o componente móvel da aplicação se encontra.

Na perspectiva do ISAM, e conseqüentemente do EXEHDA, identificar o contexto de modo a, dinamicamente, promover adaptações tanto na aplicação quanto no próprio *middleware*, é um dos requisitos básicos a serem providos por um ambiente de execução destinado à Computação Pervasiva.

Os quatro focos motivacionais, resumidos nesta seção, orientaram as decisões tomadas na concepção do EXEHDA. Este foi definido como um novo *middleware* que faculta ao usuário a execução de aplicações a partir de diferentes localidades físicas, comportando arquiteturas distribuídas de elevada abrangência e compostas por recursos de capacidades variadas, e heterogêneos quanto à plataforma de software básico que empregam.

1.3 Histórico do trabalho

Cinco projetos de que o autor vem participando influenciaram a concepção do EXEHDA. Dois são responsáveis pela origem da proposta: Opera [OPE 99] e Appelo [APE 2000], e três pelo momento atual do trabalho: Holoparadigma [HOL 2000], ISAM [ISA 2002] e contextS [CTX 2003].

O Projeto Opera [GEY 92, OPE 99] teve como objetivos (*i*) oferecer uma alternativa para simplificar a programação de máquinas paralelas e (*ii*) aumentar o desempenho da programação em lógica. Teve sua origem no Laboratório de *Génie Informatique* (Universidade de Joseph Fourier em Grenoble/França) [BRI 90, BRI 90a], com diversas atividades já desenvolvidas no Instituto de Informática da UFRGS envolvendo a exploração do paralelismo implícito na Programação em Lógica. A premissa é liberar o programador de explicitar o paralelismo disponível na aplicação. Deste modo, no OPERA, diferentes níveis de paralelismo podem ser explorados a partir de um mesmo código Prolog, desde uma execução totalmente seqüencial, até outra com a máxima concorrência possível, considerando a aplicação e a disponibilidade de recursos. Dentre os trabalhos realizados têm-se: [WER 94, BAR 96, CAS 97, COS 97, VAR 98, CEN 99], destacando-se a participação do autor nos trabalhos [YAM 93, YAM 94a, YAM 94, YAM 2000a].

É importante ressaltar que o paradigma declarativo da Programação em Lógica faculta uma clara separação entre a semântica da linguagem e o controle da execução. Esta característica é explorada no projeto OPERA, permitindo que o gerenciador da arquitetura possa decidir as ações tocantes ao controle físico da execução de forma adaptativa aos recursos computacionais disponíveis (processadores, rede, etc.). Esta visão se intersecciona com as definições da Programação Orientada a Aspectos onde o programador, de modo independente, especifica: (*i*) um código (*ii*) e os aspectos que vão dirigir as decisões do gerenciador da execução. Este comportamento adaptativo é fundamental na proposta do EXEHDA e esteve presente nos trabalhos [YAM 94a, YAM 2002a].

As atividades do OPERA entre 1996 e 1998 estiveram inseridas no contexto do projeto APPELO: Ambiente de Programação Paralela em Lógica [GEY 99, APE 2000]. Este projeto multi-institucional foi financiado pelo PROTEM III do CNPQ e envolveu cinco universidades:

- Universidade Federal do Rio Grande do Sul - UFRGS;
- Universidade Federal do Rio de Janeiro - UFRJ;
- Universidade Católica de Pelotas - UCPel;
- Universidade do Porto - UP, Portugal;
- New Mexico State University - NMSU, USA.

As idéias trabalhadas no Projeto APPELO, relacionadas com a exploração do paralelismo implícito no código de forma automática, num ambiente em que não existe previsibilidade dos recursos que estarão disponíveis, nortearam as motivações de pesquisa que conduziram ao EXEHDA.

Por sua vez, dois projetos iniciados mais recentemente: Holoparadigma [HOL 2000] e ISAM [ISA 2002] integram o escopo de atuação dos trabalhos pertinentes ao EXEHDA. Estes projetos tiveram financiamento da FAPERGS em 2000 e 2001 respectivamente. O Projeto ISAM integra as contribuições das frentes de pesquisa em andamento. A caracterização da integração do projeto Holoparadigma com o projeto ISAM é ilustrada na figura 1.3, e pode ser resumida por:

- Holoparadigma: define a base do modelo de programação distribuído utilizado [BAR 2002];

- ISAMadapt: especifica as abstrações para expressar, em tempo de desenvolvimento, a adaptação ao contexto em aplicações móveis distribuídas direcionadas para a Computação Pervasiva. Dentre outras, ISAMadapt define abstrações para definir: adaptadores para os códigos das entidades de modelagem da aplicação; políticas de adaptação que orientam a tomada de decisão do *middleware*; e elementos de contextos para nortear os mecanismos adaptativos do EXEHDA [AUG 2004];
- EXEHDA: propõe a arquitetura dos mecanismos para coordenação, comunicação e adaptação na execução das aplicações, na perspectiva da semântica *siga-me* da Computação Pervasiva [YAM 2003].

Desta forma, o Holoparadigma contempla, de forma intrínseca, questões de mobilidade lógica e distribuição. O ISAMadapt estende o Holoparadigma para prover suporte à mobilidade física e adaptação ao contexto no tocante à linguagem de programação. O código ISAMadapt é compilado para Java potencializando aspectos de portabilidade. A aplicação ISAMadapt é gerenciada pelo EXEHDA, que viabiliza um comportamento ativo e reativo na gerência das entidades de modelagem da aplicação. O *middleware* contempla o requisito de elevada escalabilidade, e suporta cooperação com as definições feitas em tempo de desenvolvimento, através do ISAMadapt, no momento de executar os procedimentos adaptativos. Estes procedimentos adaptativos têm como núcleo um mecanismo de gerência totalmente integrado ao controle da execução distribuída.

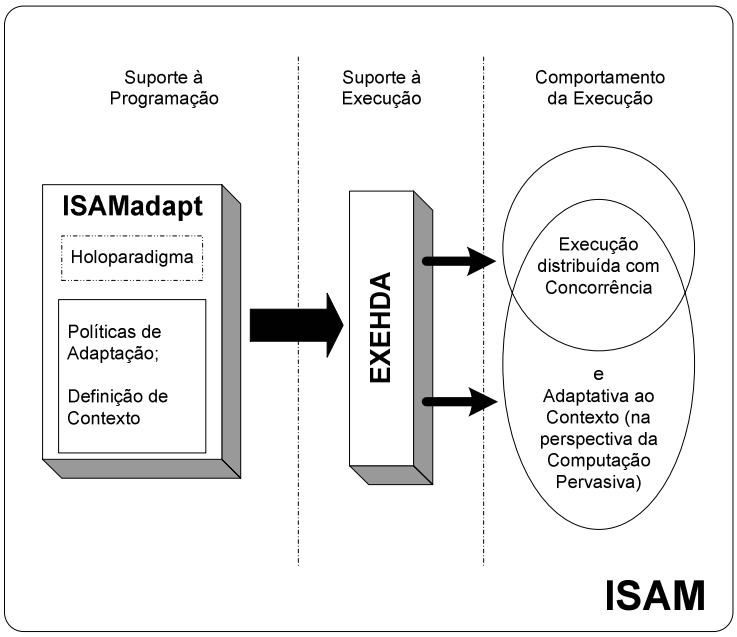


Figura 1.3: Visão geral do projeto ISAM

Por último, em fevereiro de 2003 foi iniciado o projeto *contextS*, financiado pelo CNPQ/FINEP/SEPIN, o qual tem por objetivo especializar os mecanismos relativos à monitoração e à gerência do contexto em que são submetidas as aplicações da Computação Pervasiva. Os trabalhos do *contextS* irão contribuir para especialização do

mecanismo de sensibilidade ao contexto (*context-aware*) disponibilizado pelo EXEHDA.

1.4 Objetivos

O objetivo geral desta tese é a proposição de um *middleware* para execução de aplicações da Computação Pervasiva, comprometido com as premissas do Projeto ISAM. Destacam-se como objetivos específicos:

- caracterizar as frentes de pesquisa relativas à proposição de *middlewares* para a Computação Pervasiva;
- definir princípios que devem nortear a concepção do EXEHDA enquanto *middleware* para Computação Pervasiva;
- propor um modelo computacional para o EXEHDA, que possa servir de base para implementação de modelos computacionais de mais alto nível direcionados à Computação Pervasiva;
- definir o mapeamento das construções da linguagem ISAMadapt para chamadas aos serviços do EXEHDA;
- caracterizar o modelo de organização física e lógica, dos recursos distribuídos que irão compor o ISAM *Pervasive Environment* - ISAMpe;
- definir os fundamentos para um modelo de gerenciamento do ambiente pervasivo (ISAMpe), que preserve a autonomia das instituições participantes sintonizado as demais funcionalidades do EXEHDA;
- prover estratégias e ferramentas que viabilizem o modelo de gerenciamento proposto;
- difundir o conhecimento pertinente à área de suporte à execução de aplicações na Computação Pervasiva, sobretudo no que diz respeito aos aspectos de distribuição, mobilidade física e lógica dos componentes das aplicações e sensibilidade ao contexto;
- fornecer subsídios para a elaboração de relatórios, artigos e trabalhos futuros, relacionados com o tema pesquisado.

O tratamento destes objetivos está inserido na estrutura do texto comentada na seção a seguir.

1.5 Referências do EXEHDA na WEB

Objetivando divulgar informações e coordenar as ações dos integrantes do projeto, que participam de atividades relativas ao desenvolvimento de software relacionado ao EXEHDA, foi organizado um sítio na Internet. A URL deste sítio é *http://www.inf.ufrgs.br/~exehda*. O sítio disponibiliza informações sobre as linhas de trabalho do EXEHDA, e aponta para os mecanismos e cuidados a serem seguidos na geração de código, bem como apresenta as principais publicações desenvolvidas durante a realização da pesquisa.

O símbolo do EXEHDA, apresentado na figura 1.4, procura caracterizar o seu foco de aplicação: sistemas distribuídos compostos pela interconexão de equipamentos segundo uma organização celular.

Figura 1.4: Símbolo do EXEHDA

1.6 Estrutura do texto

O texto é composto por oito capítulos e três apêndices. O uso de apêndices tem por finalidade contribuir para a objetividade da leitura e ao mesmo tempo resguardar a disponibilidade dos principais estudos complementares que foram organizados durante as pesquisas referentes ao EXEHDA. A relação entre as partes do texto está caracterizada na figura 1.5.

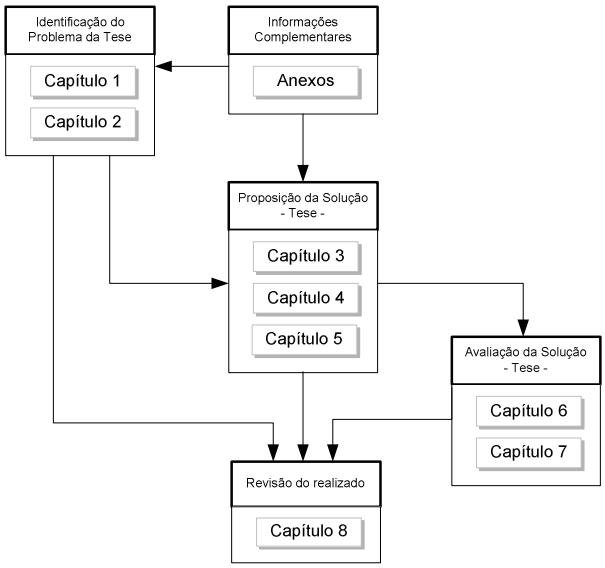


Figura 1.5: Estrutura do texto

O capítulo 2 aborda os trabalhos relacionados ao tema desta tese. Salienta-se que, no início dos trabalhos desta proposta, não eram conhecidas pesquisas que abordavam o tema infra-estrutura para a Computação Pervasiva. Nesta época, somente alguns aspectos envolvidos no conceito de *pervasividade*, como agentes móveis e monitoração do ambiente, eram abordados isoladamente. Esta situação ainda se mantém na maioria das pesquisas, e neste capítulo faz-se uma varredura das principais questões sendo tratadas. O capítulo 3 apresenta os principais requisitos e conceitos empregados na concepção do *middleware* proposto. O capítulo 4 descreve o modelo adaptativo baseado em serviços concebido para o *middleware*. Neste capítulo, estão relacionados os principais serviços, os quais estão estruturados em subsistemas que abordam funcionalidades relativas (*i*) ao acesso pervasivo; (*ii*) à distribuição; (*iii*) à comunicação; (*iv*) à adaptação; (*v*) ao reconhecimento de contexto. No capítulo 5 é relatado o suporte à

aplicação ISAMadapt. No capítulo 6 são descritas as ferramentas para o gerenciamento do ambiente pervasivo proposto. As aplicações e experimentos realizados para caracterizar a viabilidade do modelo proposto são apresentados no capítulo 7. Por fim, as conclusões, restrições da implementação do modelo e os trabalhos futuros previstos são discutidos no capítulo 8.

2 EM DIREÇÃO À EXECUÇÃO DE APLICAÇÕES DA COMPUTAÇÃO PERVASIVA

I keep six honest serving men. They taught me all I knew. Their names are:
What, Why, When, How, Where and Who.

- Rudyard Kipling

O emprego intensivo da mobilidade física – Computação Móvel – é algo recente, e mais atual ainda é seu uso integrado com redes estruturadas de recursos multi-institucionais de grande abrangência – Computação em Grade. Estas áreas têm despertado o interesse de diversos grupos de pesquisa. Porém, propostas similares ao ISAM, de integrá-las, considerando que as aplicações serão conscientes do contexto, não foram identificadas na literatura. Particulariza-se o mesmo no que diz respeito aos aspectos abordados pelo EXEHDA, seu *middleware* adaptativo. Este capítulo apresenta o estado da arte de diferentes esforços de pesquisa que se relacionam com as questões envolvidas nesta tese.

2.1 EXEHDA: contexto de pesquisa

Quando do início dos trabalhos desta tese, três iniciativas caracterizavam os esforços de pesquisa nacionais envolvendo mobilidade do usuário e/ou software: SIDAM, SIAM e ISAM, conforme quadro 2.1.

Sigla	**Nome**	**Foco**	**Universidade e URL**
SIDAM	Sistema de Informação Distribuída com Agentes Móveis	Mobilidade de código & Sistemas de Informação distribuídos	USP http://www.ime.usp.br/~sidam
SIAM	Sistemas de Informação com Agentes Móveis	Redes ad-hoc & Sistemas de Informação	UFMG http://www.dcc.ufmg.br/siam
ISAM	Infra-Estrutura de Suporte às Aplicações Móveis	Adaptação ao contexto & Computação Pervasiva	UFRGS http://www.inf.ufrgs.br/~isam

Quadro 2.1: Projetos nacionais em mobilidade de usuário e/ou software

Neste quadro, fica caracterizada a proposta diferenciada do ISAM de prover subsídios para o desenvolvimento e a execução de aplicações da Computação Pervasiva com a premissa de adaptação ao contexto. Atualmente, somente o projeto ISAM continua em desenvolvimento; os projetos SIDAM e SIAM foram desativados.No contexto internacional, diversas propostas surgiram oferecendo suporte a um comportamento adaptativo das aplicações da Computação Móvel; entre estas, destacam-

se: Coda [SAT 96], Rover [JOS 97] e Odyssey [NOB 2000], que abordam aspectos dirigidos ao acesso a dados; por sua vez, Gaia [ROM 2002] e Aura [GAR 2002] são iniciativas mais recentes, já preocupadas em prover infra-estrutura para a Computação Pervasiva.

Nenhuma das propostas analisadas, porém, contempla a preocupação em disponibilizar mecanismos genéricos para uma execução adaptativa das aplicações às condições dos recursos da infra-estrutura física no momento do processamento. Mesmo soluções adaptativas *ad-hoc*, construídas especificamente para uma determinada aplicação, ficam comprometidas, pois os *middlewares* empregados perseguem a premissa dos sistemas distribuídos tradicionais de tornar o mais transparente possível ao usuário os aspectos de gerência física da execução. Esta premissa é razoável de ser empregada em ambientes distribuídos com recursos de baixa heterogeneidade, e cuja disponibilidade é previsível; porém, este não é o caso da Computação Pervasiva.

A necessidade da consciência do contexto reforça as motivações desta tese apresentadas na seção 1.2. O projeto ISAM está direcionado para redes de grande abrangência (*wide-area*) e caracterizadas por elevada heterogeneidade, tanto na natureza como na disponibilidade dos recursos. Situação esta que não pode prescindir de um comportamento adaptativo aos recursos e serviços disponíveis no momento.

Na seção 2.2, a seguir, faz-se uma análise dos *middleware*s sob três aspectos: *middleware*s distribuídos tradicionais, *middleware*s propostos para a Computação Móvel e *middleware*s propostos para a Computação em Grade.

2.2 *Middleware*s tradicionais para sistemas distribuídos

*Middleware*s tradicionais implementam as camadas sessão e apresentação do Modelo de Referência ISO/OSI e objetivam habilitar a comunicação entre componentes distribuídos. Para tal, fornecem aos programadores de aplicações abstrações de alto nível, construídas usando primitivas do sistema operacional de rede, que escondem a complexidade introduzida pela distribuição. Tecnologias de *middleware*s existentes para sistemas distribuídos têm sido construídas com a metáfora de caixa preta, onde a gerência da distribuição torna-se transparente ao usuário e ao projetista de software [BLA 98, EMM 2000].

Os *middlewares* tradicionais para sistemas distribuídos fixos podem ser classificados em três grandes categorias: *middlewares* orientados a objetos, *middlewares* orientados a mensagens e *middlewares* orientados a transações. A classificação destes *middleware*s tem por base o aspecto comunicação/interação entre componentes. Nesta seção é feita uma revisão das principais características de cada uma destas categorias e apontadas as razões pelas quais não podem ser aplicadas com sucesso na proposta da Computação Pervasiva, adotada pelo Projeto ISAM.

2.2.1 *Middleware*s orientados a objetos

Os *middlewares* orientados a objetos têm por objetivo gerenciar as transações entre objetos distribuídos. Nestes, um objeto cliente pode requisitar a execução de um processamento em um objeto servidor, o qual pode estar alojado em qualquer outro nodo do sistema distribuído. Este tipo de *middleware* teve como base a proposta de RPCs (*Remote Procedure Calls*) [STE 90], e sua forma básica de interação entre os

objetos é síncrona. Deste modo, o objeto cliente realizando uma requisição é bloqueado até que o objeto servidor tenha retornado uma resposta.

Dentre as propostas de *middleware* nesta categoria destacam-se implementações do OMG CORBA: o Orbix da IONA [SEA 97, ION 2002] e o VisiBroker da Borland [BOR 2002], o CORBA Component Model (CCM) [OMG 2002], e o Enterprise JavaBeans [MOR 99]. O emprego destas propostas em sistemas distribuídos tradicionais tem sido bem sucedido, contudo sua aplicabilidade no cenário da Computação Pervasiva é restringida, principalmente, por três fatores: (*i*) o custo computacional que lhes é característico, (*ii*) a forma síncrona de interação entre os objetos, e (*iii*) o princípio de "transparência" que norteia suas concepções, o qual não é propício à construção de mecanismos que facultem consciência do contexto de execução por parte das aplicações. Estes fatores ficam potencializados considerando a mobilidade física, pois, além das implicações em aspectos de desempenho, a sua operação síncrona pode conduzir a soluções de baixa escalabilidade. Interesses dos grupos de usuários podem promover convergência de equipamentos para uma determinada região, daí o gerenciamento da escalabilidade ser significativo neste cenário.

2.2.2 *Middleware*s orientados a mensagens

Este tipo de *middleware* provê a comunicação entre os componentes distribuídos da aplicação através da troca de mensagens. No modelo computacional que oferece, os *componentes clientes* enviam uma mensagem contendo a requisição de um serviço e seus parâmetros, para um *componente servidor* através da rede de interconexão existente.

Os *middlewares* orientados a mensagens facultam a implementação de mecanismos de comunicação assíncrona, permitindo desta maneira um desacoplamento operacional entre cliente e servidor, (aspecto requisitado pela Computação Pervasiva). Sob esta perspectiva, o cliente pode continuar o processamento assim que o *middleware* tenha aceitado sua mensagem para despachar. Por sua vez, quando o componente servidor retornar a mensagem de resposta, a mesma é gerenciada pelo *middleware* de modo que o cliente poderá coletá-la no momento mais oportuno para computação.

Como conseqüência deste comportamento, surge uma exigência de memória nos equipamentos envolvidos, pois a mesma precisa ser suficiente para acomodar todas as mensagens geradas e ainda não processadas.

Esta característica restringe o uso de *middlewares* deste tipo em dispositivos móveis (PDAs, etc.). Como exemplo de *middlewares* com esta tecnologia tem-se o Java Message Queue da SUN [JMS 2002] e o MQSeries da IBM [MQS 2002 MQS 2002a]. Neste *middleware*s, some-se a exigência de hardwares robustos, o aspecto de não ser disponibilizada nenhuma forma de consciência do contexto onde ocorre à execução, os aspectos de gerenciamento, e particularmente as mensagens são tratadas pelo *middleware* sem participação das aplicações.

2.2.3 *Middleware*s orientados a transações

Middlewares orientados a transações são prioritariamente direcionados para aplicações que manipulam bases de dados. Sua principal característica é suportar transações envolvendo componentes que estão em execução em diferentes nodos. Um componente cliente agrupa em uma transação diversas requisições. O *middleware* se

encarrega de direcionar as transações entre os componentes servidores que se encontram distribuídos de forma transparente para ambos, clientes e servidores.

Este tipo de *middleware* se notabiliza pela sua elevada confiabilidade, porém, a garantia de atomicidade necessária no gerenciamento das transações introduz um custo computacional significativo que na maioria das vezes não será aproveitado, pois o uso de transações nem sempre se faz indispensável no cenário da Computação Pervasiva.

Novamente, os aspectos de: (*i*) carga computacional introduzida, e (*ii*) transparência operacional que inviabilizam a construção de mecanismos para sensibilidade ao contexto, tornam inadequado o uso de *middlewares* pertencentes a esta classe na construção de aplicações da Computação Pervasiva. Ressalte-se que a mobilidade física irá implicar que, na maioria das vezes, os nodos irão praticar conexões de rede não permanentes, o que aumenta a complexidade no tratamento das transações. Como exemplos deste tipo de *middleware*, destacam-se o CICS da IBM [CIC 2002] e o Tuxedo da BEA [TUX 2002].

2.2.4 Análise

A classificação do *middleware* para sistemas distribuídos sem mobilidade física (de hardware) nas três categorias abordadas, isto é, orientado a objetos, orientado a mensagens e orientado a transações, naturalmente não é rígida. Observa-se uma tentativa de fundir características destas categorias, como exemplo tem-se o *CORBA Object Transaction Service*, o qual é uma convergência da proposta orientada a objetos e da orientada a transações. Estas combinações, via de regra, levam a *middleware*s mais complexos e de maior custo computacional, o que os tornam ainda menos apropriados para um cenário de elevada heterogeneidade de recursos como o da Computação Pervasiva.

As primitivas de interação, tais como transações distribuídas, requisições a objetos ou chamada remota de procedimento, assumem uma elevada largura de banda e conexão de rede permanentemente disponível. Isso contrasta com as características inerentes ao ambiente móvel. Segundo; *middleware*s orientados a objetos, como CORBA e Java-RMI, suportam principalmente comunicação ponto-a-ponto, requerendo co-existência temporal entre cliente e servidor. Esta situação, contrasta com o ambiente móvel o qual requer comunicação anônima e assíncrona. Terceiro; sistemas distribuídos tradicionais assumem que o ambiente de execução é estacionário, deste modo com largura de banda alta, com nodos de localização fixa e serviços bem conhecidos. Esta visão contrasta com o cenário altamente dinâmico proposto pela Computação Pervasiva, onde a localização dos nodos pode alterar-se no tempo, e novos serviços podem ser descobertos dinamicamente em função disto. Finalmente, a carga computacional para execução de *middleware*s, como CORBA, é demasiadamente alta para ser carregada e executada em nodos móveis.

Outro aspecto importante é relativo à questão transparência x consciência. *Middleware*s para sistemas distribuídos tradicionais são construídos utilizando abordagens que enfatizam a transparência, onde os programadores não necessitam conhecer nenhum detalhe sobre o recurso do qual o serviço é requerido. Enquanto que, em sistemas distribuídos para nodos estacionários isto é possível, e muitas vezes desejável, ocultar completamente as informações de contexto da aplicação, em ambientes de Computação Pervasiva ante a presença da mobilidade, isto se torna mais difícil e inadequado. Para oferecer transparência, os *middleware*s tomam decisões em nome das aplicações, sacrificando a flexibilidade. No entanto, considerando as novas

demandas das aplicações móveis é mais eficiente que as decisões sobre utilização dos recursos levem em conta informações específicas (*i*) de cada aplicação, (*ii*) do nodo móvel, e (*iii*) do ambiente de execução corrente. Em sistemas móveis é essencial que o *middleware* seja adaptativo. Os *middleware*s existentes não provêem suporte a consciência do contexto, pois foram projetados com a premissa de ocultar o contexto do usuário e/ou programador.

Logo, o corrente estado-da-arte dessas plataformas fornece paradigmas de programação orientados à conexão, refletindo sua origem nas redes fixas. As aplicações móveis, no entanto, requerem um paradigma de programação assíncrono, não orientado a conexões, com suporte generalizado para controle e monitoramento de recursos.

Deste estudo, conclui-se que *middlewares* desenvolvidos para sistemas distribuídos tradicionais não são adequados para suportar o desenvolvimento de aplicações da Computação Pervasiva. Fica evidente que estes *middlewares* introduzem: (*i*) carga computacional elevada, e/ou (*ii*) suportam somente paradigmas de comunicação síncronos, e/ou foram projetados tendo em mente o (*iii*) princípio de garantir uma transparência sobre o estado da infra-estrutura de recursos em que está ocorrendo a execução.

Em face desta situação, novos *middleware*s estão sendo propostos na perspectiva de atender os requisitos da Computação Pervasiva. Têm-se duas frentes de pesquisa: (*a*) alteração dos *middleware*s existentes para suportar novos requisitos; (*b*) novas propostas de *middleware*s. Por sua vez, estudos têm mostrado que o esforço de modificação de *middlewares* já existentes para sistemas fixos, para atender a proposta da mobilidade, não é oportuno [ROM 2000, AUG 2001, YAM 2001]. Isto faz com que tecnologias de *middleware* que tiveram seu uso consolidado nos últimos anos na construção de sistemas distribuídos encontrem fortes obstáculos para serem reaproveitadas para uso em um meio de execução permeado pela alta heterogeneidade, e com a possibilidade da mobilidade física.

Salienta-se que os *middleware*s que têm sido propostos, em geral, abordam somente um ou outro aspecto necessário para a construção da infra-estrutura para a Computação Pervasiva: comunicação, adaptação/reconfiguração dinâmica, descoberta de recursos, aquisição e tratamento das informações de contexto, conforme apresentado nas seções de 2.3 a 2.8, a seguir. A arquitetura proposta para o *middleware* EXEHDA é uma seleção de serviços, cuja integração persegue aspectos de otimização, e que visa fornecer de forma integrada, toda infra-estrutura necessária para o ambiente pervasivo. Diferenciando-se, portanto, das abordagens vigentes.

2.3 *Middleware*s para comunicação

Nesta categoria estão as propostas que são revisões dos *middleware*s tradicionais, como CORBA e RMI, e novas propostas baseadas no conceito de Espaço de Tuplas, que tentam resolver o problema da desconexão e comunicação assíncrona de forma natural.

2.3.1 Revisões de CORBA

O emprego da tecnologia CORBA (*Common Object Request Broker Architecture*) ao ambiente móvel exige algumas considerações: (*a*) a arquitetura CORBA foi concebida para o ambiente fixo, considerando conexão permanente via TCP/IP, banda larga, e alta carga computacional para o processamento dos ORBs; (*b*) os aspectos enfocados são a

interoperabilidade entre aplicações de diversos fabricantes, e a comunicação entre objetos remotos; (*c*) a comunicação cliente-servidor é síncrona e altamente acoplada, os dois devem estar conectados ao mesmo tempo; (*d*) a interoperabilidade privilegia a facilidade de implementação de protocolos IIOP (*Internet Inter-ORB Protocol*) e não a otimização de uso da largura de banda do meio; (*e*) a abordagem da arquitetura privilegia a transparência da comunicação e interoperabilidade.

Desta forma, sua adoção na Computação Pervasiva exige um reprojeto da solução para os requisitos da mobilidade e da elevada heterogeneidade dos recursos de processamento. É necessário reduzir a carga computacional para o gerenciamento da comunicação e interoperabilidade. A comunicação síncrona deve ser substituída por um mecanismo com desacoplamento temporal e espacial. Deve ser introduzido suporte à desconexão, o que mudaria o protocolo de interoperabilidade (IIOP) atual. É necessário adicionar também novos serviços de localização a entidades móveis e de migração. Além da necessidade de disponibilizar informações de contexto para as aplicações.

Algumas propostas de adequação de CORBA à Computação Móvel estão sendo estudadas, entre elas o projeto DOLMEN [DOL 2003] e ALICE [ALI 2003]. Estes visam o suporte transparente à mobilidade e comunicação, e trabalham com o conceito de rede infra-estruturada. Para tratar as características do meio sem fio, as soluções incluem o gerenciamento da conectividade, e da localização dos nodos.

2.3.2 *Middleware*s baseados em Espaço de Tuplas

O espaço de tupla é uma memória associativa disponibilizada de forma global, utilizada pelos processos para se comunicarem [GEL 85]. Atua como um repositório de dados estruturados, denominados tuplas, que de modo geral podem ser entendido como um vetor de valores associados a tipos. As tuplas são anônimas, sendo acessadas através de mecanismo de casamento de padrões. Por sua vez, as comunicações não prescindem nem de acoplamento temporal, nem de espacial: o processo receptor e transmissor não precisam estar operacionais simultaneamente, pois as tuplas têm o seu tempo de validade de forma independente do processo que as gerou. Por sua vez, a localização física dos processos envolvidos também não é significativa em função da premissa de acessibilidade global do espaço de tuplas.

Como exemplo deste tipo de *middleware* têm-se [BAY 97, DAV 97, PIC 2001]. As comunicações sem fio necessárias para um efetivo suporte à mobilidade física têm os comportamentos característicos de banda-passante variável, e de desconexões freqüentes. Estes comportamentos conduzem a um modelo desacoplado e oportunístico para as comunicações: desacoplado no sentido que a computação deve prosseguir mesmo durante o período de desconexão, e oportunístico no momento que a conectividade deve ser explorada sempre que possível.

Os desacoplamentos temporal e espacial assumem enorme importância quando da mobilidade de hardware e/ou software, pois na mesma, as partes envolvidas mudam seu perfil de comunicação dinamicamente, devido a sua migração física, ou a mudanças nas condições de conectividade. É importante ressaltar que uma implementação tradicional de espaço de tupla não é suficiente. Diversas novas questões surgem na perspectiva de uso de espaço de tupla no cenário da Computação Pervasiva, dentre outras: como viabilizar com desempenho um acesso global aos dados; como construir escopos de localidade para as informações sem comprometer a conotação pervasiva; como garantir a indexação de acesso aos dados considerando a mobilidade de hardware e software.

Para tentar responder a essas questões, alguns projetos estão em andamento. Dentre estes, citam-se: LIME (*Linda in a Mobile Environment*) [PIC 2001] e TOTA (*Tuples On The Air*) [MAM 2003]. No modelo Lime, agentes móveis são programas que podem viajar entre nodos móveis e toda comunicação é via um Espaço de Tuplas Transiente, que incorpora o conceito de mobilidade (física e lógica). O Espaço de Tuplas é permanentemente associado aos agentes móveis nos nodos móveis. O compartilhamento transiente permite a reconfiguração dinâmica de seu conteúdo de acordo com a migração do agente, ou variações na conectividade decorrentes da entrada/saída de nodos móveis na rede.

TOTA estende o modelo LIME. Agentes de aplicações interagem em TOTA totalmente desacoplados, via o espaço de tuplas em uma rede ad-hoc (espontânea). Tuplas são definidas em termos de conteúdo e regras de propagação. Estas podem ser injetadas na rede e propagadas conforme o padrão especificado. Cada agente local que executa o *middleware* nos nodos móveis é capaz de armazenar e difundir tuplas. O *middleware*, escrito em Java, suporta propagação adaptativa: o conteúdo da tupla pode alterar-se durante a propagação de acordo com o estabelecido nas regras de propagação. Conforme as condições de rede monitoradas formam uma nova topologia, o *middleware* pode automaticamente re-propagar tuplas. Desta forma, tuplas são sensíveis ao contexto. Aplicações são programadas usando agentes móveis MARS [CAB 2002] e as primitivas de programação TOTA: `inject(tuple)`, `read(template)`, `delete (template)`, `subscribe(template,reaction)`, `unsubscribe (template)`.

2.3.3 Outras abordagens

O projeto Jedi [CUG 2001] investiga o impacto da mobilidade no projeto e implementação de um *middleware* baseado no paradigma *publish-subscribe*. Este paradigma foi projetado para redes fixas, considerando permanentemente disponível o acoplamento entre cliente e servidor. Portanto, não leva em conta os problemas introduzidos pela Computação Pervasiva, associados a reconfiguração dinâmica da topologia da rede, às restrições do ambiente e à mobilidade física dos nodos. Os *middleware*s existentes consideram que (*i*) *publishers* e *subscribers* são estacionários, e (*ii*) o padrão de comunicação é fixo. Para abordar esses problemas o projeto Jedi adota uma topologia infraestruturada para a rede móvel. As aplicações são organizadas como um conjunto de componentes autônomos que interagem através de notificação de eventos. Componentes podem inscrever-se em uma ou mais classes de eventos, expressando seu interesse em recebê-los. O *middleware* inclui um componente especial, o *dispatcher*, responsável por gerenciar a distribuição das notificações de eventos para todos os componentes inscritos. Para garantir escalabilidade, servidores *dispatcher* são organizados em uma estrutura de árvore, e eventos são roteados através da árvore desde o gerador até os consumidores do evento. Um protocolo de coordenação entre eles, que gerencia a propagação de eventos fora dos limites de um único servidor *dispatcher*, garante a minimização do tráfego na rede. Jedi suporta desconexão e reconexão dos componentes ao servidor *dispatcher*.

Outra direção de pesquisa na área focaliza a construção de *middleware*s para suporte ao paradigma de comunicação par-a-par (*peer-to-peer computing*) [CUG 2001a]. Neste modelo, os usuários compartilham de forma transparente com seus pares (nodos conectados), as informações disponíveis no seu nodo local, sem necessidade de publicar as dados em um servidor específico. Informações e serviços não têm mais um ponto de concentração na rede; em vez disso, cada nodo é responsável por uma parte do conjunto

de informações. Esta abordagem viabiliza a solução do problema de escalabilidade, e é indicado para o cenário altamente distribuído das redes *ad-hoc* (vide apêndice 1.1.1). Por outro lado, a manutenção de informações globais neste tipo de *middleware* apresenta um custo elevado.

2.3.4 Análise

Embora permitindo comunicação assíncrona de uma maneira muito natural, sistemas baseados em Espaços de Tuplas falham no suporte à consciência do contexto. Uma desvantagem adicional desses sistemas é em termos de capacidade de sincronização. Espaço de tuplas é multitarefas, o que significa que todas as tuplas podem ser duplicadas no espaço. Sempre que dois ou mais dispositivos, que replicam uma parte dos dados (que estão organizados na forma de tupla), desconectam-se e modificam localmente os mesmos, o processo de reconciliação do reagrupamento das tuplas durante a reconexão torna-se uma operação a ser tratada (não natural). Independentemente deste aspecto, este modelo de comunicação e sincronização é considerado oportuno para o ambiente da Computação Pervasiva, e foi adotado pelo EXHEDA.

2.4 *Middleware*s para adaptação

Destacam-se nesta categoria as propostas que tentam construir *middleware*s leves e reconfiguráveis, usando o conceito de reflexão computacional, ou que objetivam fornecer suporte para a reconfiguração dinâmica das aplicações.

2.4.1 *Middleware*s baseados em reflexão

O papel da reflexão nos *middleware*s é trazer maior acesso e flexibilidade na interação com os mesmos, uma vez que um dos problemas para sua adoção na Computação Pervasiva decorre do fato que um *middleware*, enquanto plataforma intermediária, abstrai os detalhes e as decisões de mais baixo nível com relação à aplicação. Um sistema reflexivo permite modificar o *middleware* através da inspeção e da adaptação. Com a inspeção, o comportamento interno é exposto, sendo possível inserir comportamentos específicos para monitorar a implementação do *middleware*. Através destes mecanismos, fica viabilizada a disponibilização de informações de contexto para a aplicação e para os mecanismos de gerência do próprio *middleware*. Através da adaptação, em decorrência das informações de contexto, o comportamento interno do próprio *middleware* pode ser alterado dinamicamente.

OpenORB [CLA 2001], OpenCorba [LED 99], dynamic TAO [KON 2000], MULTE-ORB [ELI 2002], Flexinet [FUH 2002] e Globe [STE 99], são exemplos de *middlewares* que estendem *middleware*s tradicionais, como CORBA, e abordam a adaptação através do conceito de reflexão computacional. OpenCorba [LED 99] é um *broker* reflexivo que habilita os usuários a adaptar a representação e a execução das políticas do software dinamicamente. É possível aos objetos mudarem dinamicamente suas classes em tempo de execução, para, por exemplo, melhorar suas implementações. O mesmo pode ser dito para classes e metaclasses: a classe de uma classe (i.e, a metaclasse), pode ser mudada dinamicamente, de tal forma que as propriedades podem ser adicionadas e removidas durante a execução, sem uma recompilação integral do código.

Aspectos reflexivos do OpenCorba são essencialmente baseados na habilidade de manusear mensagens enviadas. Por exemplo, considere-se a invocação de um objeto, como ocorre em uma implementação CORBA padrão: no lado do cliente há uma representação local do servidor de objeto, chamado objeto Proxy. A classe Proxy tem associada uma metaclasse Proxy cuja tarefa é interceptar cada requisição dirigida para o remote Server object, roteá-la para o servidor real, e então voltar o resultado para o cliente. Desejando-se mudar este mecanismo de expedição, por exemplo, para permitir ao objeto servidor a migração para o lado do cliente para otimizar o desempenho, uma metaclasse diferente pode ser associada ao objeto servidor em tempo de execução.

Como já visto na seção 2.2.1, as implementações CORBA padrão, em função do custo computacional que introduzem, não são adequadas para processar em equipamentos móveis. Adicionando-se a esse fato o problema da indireção introduzido pela reflexão, este *middleware* tende a apresentar um custo computacional maior. Este mesmo raciocínio aplica-se aos outros *middleware*s reflexivos estudados: OpenCorba, dynamic TAO, MULTE-ORB e Flexinet.

2.4.2 *Middleware*s para reconfiguração dinâmica

Diversas adaptações são possíveis na busca do balanço entre custo e benefício para o usuário. Técnicas mais usuais incluem compressão, criptografia, filtragem, priorização, *prefetching*, *caching* e bufferização. Em adição, propriedades específicas de protocolos de apresentação de dados, por exemplo, podem ser alterados. Propostas existentes freqüentemente instalam *proxies* que filtram e/ou comprimem dados para uma aplicação específica. Esses filtros podem ser habilitados ou desabilitados. Essas soluções são construídas de forma específica, como exemplos neste sentido estão às aplicações multimídia adaptativa [NOB 2000].

A maioria dessas pesquisas é baseada na premissa que mobilidade deve ser escondida dentro de níveis de abstração. Assim, em princípio, permitindo as aplicações e serviços existentes executarem com um mínimo de modificações. No entanto, suporte a adaptação na Computação Pervasiva é requerido em todos os níveis desde a infra-estrutura básica de comunicação, no *middleware* como um todo e na aplicação.

Sistemas de reconfiguração dinâmica, como DACIA [LIT 2001], abordam questões relativas à alteração da estrutura da aplicação, inserção, alteração ou remoção de componentes durante o processamento. DACIA destina-se à construção de aplicações *groupware* configuráveis em um ambiente móvel. Suas características são: (*a*) o usuário pode "pegar" (*pull)* um componente de um dispositivo e "colocar" (*drop)* em outro; (*b*) os componentes que implementam funções individuais podem trocar sua estrutura em durante a execução (*runtime)*: carregar novo subcomponente, trocar a forma de interação dos subcomponentes, trocar dados, mover as funções (subcomponentes) de um nodo para outro e replicar algumas funções entre os nodos; (*c*) a aplicação pode ficar estacionada em um nodo quando o usuário está desconectado ou ocioso. Neste caso, a aplicação pode continuar, com algumas restrições, a interagir com outras partes em nome do usuário.

2.4.3 Análise

A característica de consciência de si próprio por parte do sistema computacional, culminando na capacidade de se adaptar às condições dos recursos disponibilizados pelo meio físico de execução (contexto do meio), é necessária para o EXEHDA, pois o

mesmo visa trabalhar com (*i*) equipamentos cuja natureza abrange *palmtops*, *desktops* e *workstations* e, portanto, significativamente diferentes no tocante ao perfil computacional; (*ii*) suporte à mobilidade física e, assim, possibilitar a troca de ambiente onde acontece o processamento; e (*iii*) ambiente de execução de abrangência global, e, deste modo, sujeito a elevadas flutuações nas diferentes instâncias locais da disponibilidade de recursos.

Os sistemas reflexivos, porém, precisam ser repensados para a perspectiva da Computação Pervasiva. O processo de inspeção e adaptação, através do mecanismo de interceptação de mensagens, introduz custos e é construído sob a premissa de que os nodos estarão permanentemente conectados, o que não é verdadeiro no cenário introduzido pela pervasividade.

2.5 *Middleware*s para reconhecimento do contexto

Obter informações do ambiente é, via de regra, uma atividade complexa e cara para os sistemas computacionais [YAM 99]. Porém, no ambiente móvel, estas informações são um requisito básico para o comportamento adaptativo das aplicações. Chen [CHE 2001] faz uma revisão do uso da informação de contexto por parte das aplicações móveis. Como exemplo de possíveis aplicações de uso, ele destaca: Teleporting (localização de equipamentos vizinhos e seus serviços); Shopping Assistant e CyberGuide (orientações para pessoas que se deslocam); Object Pager (envio de avisos dependendo da localização do usuário). É importante observar que se tratam de aplicações construídas para atender de forma específica as necessidades de seu foco de aplicação, as quais prescindem de um *middleware* propriamente dito.

A premissa da Computação Pervasiva de que o usuário, ou o código da aplicação, ou ambos poderão se deslocar conduz à situação de estar a aplicação em diferentes contextos durante seu tempo de execução. Para que as aplicações possam se adaptar às condições do meio em que se encontram, o *middleware* deve prover informações sobre o contexto no qual as mesmas estão em processamento no momento. Dentre outras, as informações de contexto incluem: localização, cuja precisão depende do recurso para posicionamento utilizado; localização relativa, como a proximidade a recursos em geral: impressoras, bases de dados e servidores; características do dispositivo, como poder de processamento e recursos de entrada e saída; informações sobre o ambiente de rede, como nível de ruído e largura de banda; meio do usuário, como no automóvel, em um museu ou no ambiente de trabalho; atividade sendo executada pelo usuário.

Somente recentemente *middleware*s para fornecer informações de contexto às aplicações começam a ser propostos. Estes tratam questões relativas a obter informações sobre uma determinada entidade, como localização de uma pessoa ou objeto, ou abordam modelagens mais genéricas, que poderão dar o suporte exigido pela Computação Pervasiva.

2.5.1 Contextos específicos

Aplicações *network-aware* (conscientes da rede) devem ser capazes de obter informações sobre a disponibilidade de recursos, em particular a capacidade e o estado atual da rede [AUG 2002a]. Porém, arquiteturas de rede diferem significativamente na habilidade de fornecer tais informações para o nodo ou para a aplicação executando no nodo [ANG 98, GRO 99]. A fim de evitar dependências de idiossincrasias da arquitetura de rede e sistemas de comunicação, o desenvolvimento dessas aplicações requer uma

interface independente do sistema. Isto permite o desenvolvimento de aplicações portáveis que podem se adaptar a qualquer rede, e que podem fornecer informações adequadas. Além disso, essa interface é necessária para permitir às aplicações explorarem a heterogeneidade das redes. Em [GRO 99] é descrita a interface `Remos` para as aplicações obterem informações relevantes da rede necessárias a uma adaptação efetiva.

Por sua vez, as aplicações *location-aware* (conscientes de localização) têm como componente central do seu contexto de interesse a localização. Um exemplo de *middleware* que provê informações baseadas na localização é o Nexus [FRI 2000], o qual objetiva fornecer suporte a um ambiente heterogêneo de comunicação. Na arquitetura existem quatro componentes que operam juntos, um no cliente móvel e outros três no interior da plataforma Nexus. Estes componentes são:

a) interface do usuário, executa no cliente e contém a funcionalidade básica requerida pelo `Nexus Client` para interagir com a plataforma Nexus: mostrar e navegar através do modelo. Fornece suporte para adaptar-se ao dispositivo com diferentes níveis de recursos e conexões de rede;

b) sensores para posicionamento no ambiente interno ou externo baseados em GPS, ou informações de satélites;

c) comunicadores, permitem a conexão com a fonte de informação, via Internet e comunicação sem fio. Esta parte atua como uma ponte para diferentes tecnologias de redes: LAN sem fio, Bluetooth, telefones móveis com GSM ou UMTS, e outras;

d) gerenciadores de dados distribuídos, onde dados espaciais são fornecidos em múltiplas representações.

A restrição é que Nexus suporta somente um elemento de contexto que é a localização, limitando o domínio de aplicações que podem ser projetadas com o suporte do mesmo. Outros elementos do ambiente, como banda e bateria, também devem ser levados em conta para o desenvolvimento de aplicações de uso geral.

As informações de contextos projetadas para o projeto Aura [GAR 2002] e o projeto Gaia [ROM 2003] também são específicos. No primeiro, a entidade de contexto modelado é o usuário: suas atividades e preferências disparam o processo de adaptação automática no ambiente. No segundo, o contexto é relativo a um espaço ativo, onde objetos podem entrar e sair dele.

Chen [CHE 2001] observou que poucos contextos, além de localização, tem sido usado nas aplicações correntes. Em parte, esta situação é devida à inexistência, até então, de suporte ao reconhecimento de elementos de contexto genérico.

2.5.2 Contextos genéricos

Recentemente, pesquisadores têm se interessado em propor infra-estruturas de suporte à coleta e tratamento das informações de contexto de forma genérica, impulsionando o desenvolvimento de aplicações na área de *context-aware computing*. Dentre estes, destacam-se Context Toolkit [DEY 2000] e Solar [SOL 2002].

Context Toolkit provê um *framework* para aplicações e um *middleware* cujas funcionalidades são, coletar informações de sensores, traduzi-las e disseminá-las para as aplicações interessadas Este Toolkit vê o contexto como um conjunto de componentes

que mapeiam objetos e locais do mundo real os quais representam pessoas, locais e coisas em geral. Inspirado no conceito de interfaces gráficas, `context widgets` obtém informações de sensores, passando-as para interpretadores, que traduzem uma informação em outra, ou para servidores de agregação, que compões informações gerando outra. Cada `widget` tem um estado, composto por um conjunto de atributos, e um comportamento expresso por um conjunto de funções `callback` disparadas pela troca no contexto (notificação de evento). O foco do Context Toolkit é em aplicação interativas que detectam a presença de pessoas ou objetos num lugar e reagem a esta presença. O uso dessas informações deve ser tratado pelos componentes da aplicação, `widgets`, como sugerido pelo *framework*. Outro problema é relativo ao mecanismo de comunicação implementado: (*i*) que força os componentes terem conhecimento da localização dos demais, (*ii*) é baseado num protocolo de rede fixa inadequado ao ambiente móvel.

O projeto Solar propõe uma abstração baseada em grafos para a agregação e disseminação da informação de contexto. A estrutura é dirigida por eventos; eventos representam as trocas de contexto. A fonte de informação de contexto são os publicadores de eventos (sensores), e as aplicações subscrevem-se para receber os eventos de seu interesse. Para reutilizar funções ou subfunções de agregação de informações de contexto entre aplicações é definido o conceito de grafo de operadores. Operadores pertencem às categorias: filtros, transformadores, unificadores ou agregadores, e são disponibilizados como objetos Java que implementam estratégia `publish/subscribe`. Cada aplicação define sua árvore de fontes de informação e operadores em uma linguagem baseada em XML. Para reusar a informação de contexto, a cada requisição de subscrição, Solar verifica se já existe uma subárvore definida e subscrita por alguma aplicação, fazendo o compartilhamento de subscrição. No sistema Solar, uma `Star` centralizada processa as requisições de subscrições das aplicações; um `Planet` é uma plataforma de execução para fontes e operadores, responsável por rastrear subscrições e enviar eventos no grafo de operadores. `Planets` periodicamente registram-se na `Star`. Quando um novo operador deve ser instanciado, a `Star` decide em qual `Planet` este será criado. Aplicações executam fora do sistema Solar, e utilizam a biblioteca fornecida com a finalidade de prover interfaceamento com o sistema. Solar também fornece um *framework* para programação de aplicações.

2.5.3 Análise

A maioria dos *middleware*s que disponibilizam reconhecimento do contexto interage diretamente com a camada de rede do sistema operacional para obter informações, processá-las e, então, apresentá-las de uma forma conveniente para o usuário. Sua principal característica é a solução personalizada assumida para cada equipamento e/ou conjunto de sensores. Estas soluções *ad-hoc* dificultam o aproveitamento do código da aplicação e, sobretudo, a reutilização das soluções adotadas pelo *middleware* para suporte à execução.

Por outro lado, as soluções genéricas adotam o modelo *publish-subscribe* para enviar eventos, que representam a troca de contexto, para as aplicações interessadas. Estes também impõem o uso de *framework*s para o projeto de aplicações. Neste caso, muito trabalho fica a cargo do programador. Além disso, questões como desconexão do cliente ou disseminação eficiente das informações não são abordadas. O modelo de definição do contexto é outro foco de atenção. Portanto, ainda existem muitos pontos em aberto que precisam ser pesquisados.

2.6 *Middleware*s para gerenciamento de recursos

Recursos e serviços tornam-se disponíveis/indisponíveis à medida que o tempo transcorre e usuários se deslocam. O gerenciamento desses recursos é um aspecto essencial às aplicações que necessitam destes para operar. Devem ser investigados mecanismos para descoberta, recuperação e associação dinâmica de recursos disponíveis, considerando a localidade em questão, e que o cliente tem conhecimento incompleto sobre a existência destes.

2.6.1 Descoberta de recursos

*Middleware*s para descoberta de recursos possuem duas funções-chave: anúncio e a descoberta do recurso/serviço. Para tal, os mecanismos de descoberta de recursos usam protocolos específicos. Os protocolos mais usados são:

- SLP (*Service Location Protocol*) é o padrão IETF (*Internet Engineering Task Force*) composto por três agentes: (*i*) agentes de usuário, atuam no lado cliente, e têm como principal responsabilidade a efetivação da requisição por serviços necessários à aplicação do cliente; (*ii*) agentes de diretório, atuam como agentes intermediadores. Consistem em um repositório de endereços de serviços, o qual pode ser consultado quando um serviço é solicitado; (*iii*) agentes de serviço, responsável pelo anúncio dos serviços. Este anúncio pode ser, por exemplo, uma mensagem enviada a todos os agentes de diretório disponíveis na rede, com o intuito de registrar o serviço oferecido nestes repositórios.
- JINI é uma arquitetura desenvolvida pela Sun Microsystems e implementada com a tecnologia JAVA, onde serviços podem possuir atributos, para descrever suas características, e podem operar em grupos. Desta forma, os atributos atuam como características do serviço, permitindo que um serviço seja diferenciado de outro. A operação baseada em grupos permite que se configure a rede em setores, constituídos por um conjunto de serviços. JINI opera com o conceito de *Leasing*, que consiste em um tempo de vida do serviço. Os serviços devem ser renovados antes do tempo de *Leasing* expirar. No caso de uma não renovação, a arquitetura considera que o serviço não está mais disponível e remove o seu registro. A mobilidade de código é utilizada nesta tecnologia, onde o cliente de um serviço pode fazer o download do objeto que representa o serviço e junto o código que possibilita a utilização deste serviço. A arquitetura JINI é composta por três componentes: (*i*) cliente, dispositivo ou serviço que necessita utilizar um serviço disponível na arquitetura; (*ii*) provedores de serviço, entidade que tem a capacidade de executar a solicitação de um cliente, ou seja, prestar um serviço; (*iii*) diretório (*Lookup services*), entidade que atua como um repositório de registros de serviços. Este é um mecanismo de inicialização central do sistema proporcionando o principal ponto de contato entre o sistema e os usuários do sistema;

- UPnP (Universal Plug and Play) é o padrão Microsoft para descoberta espontânea, direcionado para o mercado doméstico e de pequenas empresas. A motivação do protocolo é “configuração zero” para que novos dispositivos e periféricos sejam inseridos. É similar ao SLP, e usa um conjunto de protocolos para descoberta, interação e notificação de eventos. Um destes é o protocolo SSDP (*Simple Service Discovery Protocol*). Um serviço anuncia-se na criação e, a qualquer tempo, pode anunciar seu estado; clientes podem pesquisar por serviços quando requisitado. Ambas operações usam multicast sob IP. A resposta contendo todos os serviços encontrados é enviada diretamente ao cliente que requisitou.

Esses protocolos não são especificamente apropriados para as restrições impostas pela Computação Pervasiva, particularmente pela característica da mobilidade. Algumas pesquisas focalizam a adequação dos protocolos-padrão ao ambiente móvel, como JiniME, Jini para pequenos dispositivos como celulares e palmtops. Esta versão não apresenta chamada remota de métodos (RMI) nem carga dinâmica de código dos serviços. A solução é direcionada à comunicação dispositivo-dispositivo, redes ad-hoc, onde cada dispositivo contém o serviço de busca embutido.

Diferentemente, projetos, como MARE (*Mobile Agents Runtime Environment*) [STO 2002] e Solar [SOL 2002], examinam a possibilidade de novas abordagens. MARE trata configuração de recursos usando agentes e espaço de tuplas. O *middleware* é composto por agentes, recursos e gerenciadores. Cada nodo contém uma instância de MARE. Um `Agent Manager`, periodicamente escuta por agentes que entram no espaço de tuplas. Após a detecção do agente, seus requisitos são avaliados; se aceitos, um ambiente de execução é fornecido ao agente. Um `Resource Manager` recebe e transmite informações sobre os recursos e serviços para os agentes executando na instância de MARE. Serviços periodicamente se anunciam ao `Resource Manager`, gerando quatro mensagens no espaço de tuplas. Este mantém a visão dos recursos disponíveis no nodo.

O projeto Solar aborda este problema sob a ótica de que aplicações pervasivas devem descobrir e usar recursos com base no contexto corrente. Contexto é definido como a circunstância na qual a aplicação executa, e pode incluir estado físico, estado computacional e estado do usuário. O foco é o serviço de nomes, que permite aos recursos anunciarem-se com nomes sensíveis ao contexto, e às aplicações fazerem consultas sensíveis ao contexto. Os requisitos a serem perseguidos por esta solução são: flexibilidade, escalabilidade, rapidez para suportar freqüentes atualizações, e responsividade.

2.6.2 Alocação dinâmica de recursos

O objetivo de todo sistema computacional é maximizar o uso dos seus recursos e atender as necessidades das aplicações. O elevado nível de variação na disponibilidade de recursos, devido à mobilidade dos usuários e/ou das suas aplicações, resulta na necessidade de novos mecanismos de alocação de recursos, os quais devem apresentar um comportamento adaptativo.

Mecanismos tradicionais baseados em reserva não são apropriados, pois o sistema pode não ter condições de honrar com a reserva feita à medida que novos usuários entram na célula e requisitam recursos. O projeto Timely [BHA 98] propõe o modelo de serviços adaptativos os quais permitem as aplicações e gerente de recursos

(re)negociar qualidade de serviço (QoS) dependendo das condições dinâmicas da rede. Serviços adaptativos têm três características: (*i*) noção de faixas de QoS; (*ii*) parâmetros de QoS relacionados à mobilidade; (*iii*) classes de QoS. Neste modelo, a aplicação especifica os limites máximos e mínimos de QoS aceitáveis. O sistema garante a reserva mínima de recursos solicitados. Para permitir a (re)negociação, a aplicação é dividida em blocos de adaptação, os quais consistem de seqüências alternativas de execução, cada uma associada com uma classe de QoS. Na entrada do bloco, a aplicação negocia com o sistema a classe de QoS a ser atendida. Esta é garantida até o final da execução do bloco. Desta forma, o sistema pode redistribuir recursos entre aplicações.

O deslocamento do usuário, durante a execução de uma aplicação, introduz outro problema: como manter a associação de recursos da aplicação aos recursos físicos do ambiente (*binding*) se a aplicação ultrapassar o limite de uma célula e entrar em outra. Esta associação deve ser dinâmica, operar de acordo com as condições do ambiente, e automaticamente escolher a melhor estratégia para a situação. Em princípio, quatro estratégias podem ser usadas: (*a*) mover o recurso para a nova localização – aplicável a recursos como base de dados, requer recursos de tráfego e armazenagem no hospedeiro; (*b*) mover uma cópia do recurso, requer o tratamento de réplicas e recursos no hospedeiro; (*c*) referência remota, requer conexão e ambiente de execução remota no hospedeiro; (*d*) *rebinding*, requer localizar um recurso correspondente na nova localização. A escolha depende de vários fatores: condições do ambiente de execução, propriedades dos dispositivos-clientes, requisitos do gerenciamento de recursos, preferências dos usuários. Outra questão a ser tratada é que a estratégia de associação dinâmica, tipicamente, está embutida no código do serviço, o que limita a flexibilidade do gerenciamento consciente do contexto.

COLOMBA (*COntext and LOcation-based Middleware for Binding Adaptation*) [BEL 2003], construído no topo da plataforma SOMA [BEL 2001], suporta associação dinâmica de recursos através da exploração de metadados. Estes metadados são utilizados para descrever recursos, dispositivos, informações sobre os usuários, e estratégias preferenciais; e serviços do *middleware*. O *middleware* é estruturado em dois componentes: (1) `binder manager` - intermedia o acesso aos recursos de acordo com a política especificada; (2) `policy manager` - suporta especificação, instalação dinâmica e alocação de políticas.

2.6.3 Análise

Para uma aplicação ser capaz de executar num ambiente altamente dinâmico, é necessário um serviço de gerenciamento dos recursos que, adequadamente, trate os problemas decorrentes do movimento de usuários, dispositivos e aplicações. Este serviço deve transparentemente descobrir os recursos necessários à aplicação e dinamicamente (re)alocá-los considerando as condições e requisitos correntes do ambiente.

Para ser capaz de usar um serviço de descoberta de recursos no ambiente pervasivo, o cliente deve usar o servidor de descoberta disponível ou, em sua ausência, ele próprio ter um mecanismo que permita a descoberta autônoma. Ou seja, o mecanismo de descoberta deve ter tanto um comportamento centralizado, utilizando um serviço de diretório, quanto distribuído, permitindo descoberta par-a-par a partir do dispositivo do usuário. Outras necessidades para os protocolos de descoberta são: auto-reconfiguração; linguagem para descrição dos recursos e consultas; interoperabilidade entre protocolos e plataformas; escalabilidade; adaptação automática para mobilidade e disponibilidade

esporádica. Conclui-se que, novos protocolos devem ser propostos para atenderem a essas exigências. Além disso, uma análise mais aprofundada sobre o impacto da mobilidade nos mecanismos de gerenciamento de recursos é necessária.

2.7 *Middlewares* para a Computação em Grade

O ambiente de execução necessário ao ISAM é caracterizado por uma grande abrangência física (*wide-area*), podendo atingir uma escalabilidade no nível de Internet. Esta configuração de *network computing* tem forte semelhança com um ambiente tipo grade (*Grid Computing*) [KRA 2001].

No que diz respeito ao software, na Computação em Grade destacam-se, atualmente, duas naturezas: Grades Computacionais - seu principal objetivo é reduzir o tempo necessário para execução das aplicações; e Grades de Dados - constituído pelos sistemas que compõem informações a partir de repositórios de dados distribuídos. As atividades de pesquisa dedicadas a Computação em Grade focalizam uma das naturezas citadas, e dentro desta uma aplicação em particular, por exemplo; GridGene (Pesquisa Genômica no Grid Computacional), GriPhyN (Grid Physics Network) e PPDG (Particle Physics Data Grid). O desenvolvimento do *middleware* nestes trabalhos, apesar de contemplar alguns esforços na mesma direção do ISAM, tem como principal diferença a especificidade das suas aplicações-alvo.

Diversos trabalhos têm surgido nos últimos anos relacionados à Computação em Grade [KRA 2001]. Um destaque neste sentido é o Globus Metacomputing Toolkit. O projeto Globus [FOS 98] disponibiliza uma “grade de recursos computacionais” [FOS 99] integrando equipamentos heterogêneos em um único sistema de grande abrangência. De forma similar à proposta ISAM, ele contempla uma estrutura escalável e distribuída para o gerenciamento de recursos.

Apesar de conter módulo específico para o controle de aplicações (GEM – *Globus Executable Management Service*), a atual versão trata as aplicações como um único executável [FOS 2001], ao invés de uma coleção de componentes que podem ser dinamicamente instanciados de um repositório de código, como se propõe para arquitetura ISAM [ISA 2002, YAM 2003].

O AppLes (*Application-Level Scheduling*) é direcionado para o desenvolvimento de agentes de escalonamento para processamentos de aplicações em redes de grande abrangência (*Metacomputing*) [SHA 2000]. Ele constrói agentes para cada aplicação (caso-a-caso), os quais são responsáveis pelo mecanismo de escalonamento. O Apples utiliza o NWS (*Network Weather Service*) [WOL 99] para monitorar a variação da carga no momento de selecionar os recursos de processamento ou comunicação (rede) que serão utilizados pela aplicação. De forma similar ao ISAM, para cômputo do escalonamento, o agente considera as necessidades da aplicação juntamente com o estado dos recursos. O ISAM se diferencia no momento que emprega uma solução baseada em políticas e comandos de adaptação que configuram os serviços de adaptação, não exigindo a programação de agentes de escalonamento para cada uma das aplicações.

O NetSolve [CAS 98] é um sistema para computação distribuída que oferece recursos para que o usuário resolva problemas científicos que exijam computação numérica intensiva. É possível ao usuário acessar recursos de hardware e software distribuídos ao longo da rede. O agente NetSolve realiza o escalonamento procurando pelos recursos que oferecem o melhor desempenho. A aplicação precisa ser construída

utilizando a API fornecida pelo NetSolve para executar processamento baseado em RPC (*Remote Procedure Call*). A principal diferença em relação ao ISAM é que o NetSolve é orientado para uma proposta cliente-servidor, enquanto que o ISAM é mais flexível e contempla inclusive aspectos de mobilidade de código.

Outros sistemas como Globe [STE 99], Legion [GRI 97], e WebOs [VAH 98], apesar de suportarem diferentes níveis de configuração, não consideram a adaptabilidade ao contexto para aplicações e ambiente de execução.

De modo geral, pode-se dizer que, apesar de nos últimos anos a pesquisa na Computação em Grade ter se intensificado, diferentemente do ISAM, os trabalhos desenvolvidos não consideram a mobilidade física dos equipamentos.

2.8 *Middlewares* para criar um ambiente pervasivo

Quando do início do projeto ISAM, não era de conhecimento dos autores nenhum projeto que contemplasse uma infra-estrutura para criação, gerenciamento de um ambiente pervasivo e para programação de aplicações para este ambiente. Recentemente, projetos voltados para a fornecer um ambiente pervasivo estão surgindo: o projeto Gaia [ROM 2003] e o projeto Aura [GAR 2002].

2.8.1 Gaia

No projeto Gaia (www.cs.uiuc.edu/gaia) é defendida uma visão de futuro onde o espaço habitado pelas pessoas é interativo e programável, e chamado de espaços ativos (*Active Spaces*). Os usuários interagem com seus escritórios, casa, carros, etc... para requisitar informações, beneficiar-se dos recursos disponíveis e configurar o comportamento de seu habitat. Dados e tarefas estão sempre acessíveis e são mapeados dinamicamente para os recursos convenientes e presentes na localização corrente. Este ambiente interativo, centrado no usuário, requer uma nova infra-estrutura de software para operar com os recursos, observar propriedades do contexto, assistir o desenvolvimento e execução das aplicações [ROM 2003].

Gaia é um *middleware* experimental usado para prototipar gerenciamento de recursos e fornecer uma interface orientada ao usuário [ROM 2002]. A pesquisa limita o espaço físico para o espaço usado pelos professores: classes de aula, escritórios e salas de leitura. GaiaOS é um meta-sistema operacional que objetiva suportar e executar aplicações portáteis em espaços ativos. Gaia coordena as entidades de software e dispositivos em rede dentro dos espaços ativos, exporta serviços para pesquisar e utilizar recursos, acessar e usar o contexto corrente, e também fornece um *framework* para desenvolver aplicações.

A estrutura de Gaia tem três grandes blocos: *kernel*, *framework* e aplicações. O *kernel* é composto pelo gerenciamento dos componentes e pelos serviços fornecidos. O gerenciamento de componentes distribuídos inclui carregar, descarregar, transferir, criar e destruir todos os componentes e aplicações Gaia. Os serviços são ativados de forma particularizada à localização, contexto, eventos, repositórios com informações sobre os espaços ativos. O contexto refere-se a informações de presença de objetos e pessoas, e condições ambientais como temperatura e som. O *framework* fornece mecanismos para construir, executar ou adaptar aplicações existentes para espaços ativos (ROMAN, 2003). Este é composto por uma infra-estrutura de objetos distribuídos, um mecanismo de mapeamento, e políticas de customização das aplicações. A atual implementação de

Gaia usa CORBA e a linguagem de script LuaORB para configurar e criar espaços de objetos. Um algoritmo de *bootstrap* interpreta o arquivo de configuração, escrito em LuaORB, e inicializa os correspondentes serviços do *kernel*. A implementação atual define quatro entidades básicas: pessoas, dispositivos, serviços e aplicações. Os recursos são descritos através de suas propriedades, expressas em XML. Para demonstrar a funcionalidade de Gaia foi desenvolvida a aplicação de Gerenciamento de Apresentações [HES 2002].

2.8.2 Aura

O projeto Aura (www.cs.cmu.edu/~aura) é uma proposta recente, e visa projetar, implementar, empregar e avaliar sistemas de larga escala para demonstrarem o conceito de "aura de informação pessoal" que se espalha pelas diversas infra-estruturas computacionais. É um grande projeto que investiga novas arquiteturas para o ambiente *pervasivo*. Seu foco é no usuário, suas tarefas e preferências [GAR 2002]. Desta forma, o conceito de contexto embutido em Aura prevê a ênfase na dimensão pessoal.

Aura visa dar suporte computacional a cenários de aplicações como: "Fred está em seu escritório preparando um encontro no qual deve fazer uma apresentação e demonstração de software. A sala do encontro é a 10 minutos de onde se encontra. No horário do encontro, Fred ainda não está pronto. Ele pega seu Palm e caminha para a porta. Aura transfere o estado do seu trabalho do desktop para o Palm, e permite a ele terminar a apresentação usando comandos de voz enquanto se desloca. Aura infere onde Fred está indo com base em informações de seu calendário e seu deslocamento físico. Aura carrega a apresentação e o software de demonstração no computador de projeção, e inicializa o projetor".

A arquitetura de software Aura, para atender este cenário, é composta de:

1. observador de contexto pessoal que interpreta o contexto físico do usuário e identifica localização, antecipa o movimento do usuário e identifica o foco da atenção deste;
2. gerente de tarefas que mantém a representação da tarefa e mapeia entre contexto e preferências do usuário;
3. gerente do ambiente que conhece o ambiente computacional e pode descobrir e montar componentes para completar a tarefa, também reconhece os recursos disponíveis e avisa quando há troca no ambiente do usuário.

Estes componentes trabalham juntos para atender as tarefas de Fred. O observador de contexto reconhece que Fred está em seu escritório. O gerente de tarefas nota que sua preferência é entrada de dados via teclado. O gerente de ambiente é responsável por encontrar os componentes que fornecem tais serviços. Quando Fred começa a se deslocar (como notado pelo observador de contexto), o serviço que melhor preenche as necessidades de entrada de texto é o ditado (como notado pelo gerente de tarefas). Aura faz uma escolha: reconhece a entrada via fala no Palm, a qual tem capacidade limitada de vocabulário, e transmite a voz para um servidor remoto, que pode tornar-se indisponível conforme Fred se desloca. O gerente de ambiente escolhe outros componentes para atender Fred, caso isto ocorra.

Os desafios do projeto incluem (a) inferência de tarefas; (b) representação das tarefas; (c) alocação de recursos [GAR 2002]. O foco num novo modelo de programação é a inovação do Aura. O Projeto Aura está em estágio de proposta.

2.8.3 Análise

Gaia e ISAM têm objetivos semelhantes. Ambos são propostas de infra-estrutura, em desenvolvimento, para Computação Pervasiva, porém as soluções são diferentes. ISAM não se limita a espaços definidos, mas considera o deslocamento global do usuário, e seu gerenciamento. O contexto ISAM é mais genérico, e pode ser definido pela aplicação (programador). ISAM inclui uma linguagem, e permite desenvolver aplicações adequadas ao ambiente, enquanto que Gaia visa à adaptação de aplicações existentes.

A arquitetura ISAM é mais ampla e, de certa forma, absorve a do projeto Aura. Aura focaliza a computação centrada no usuário, enquanto que ISAM aborda a questão de projeto e execução de aplicações móveis conscientes do contexto, focalizando a infra-estrutura de software. A mobilidade física do usuário e a semântica de que a aplicação o segue é central em ambos. A semelhança maior entre os projetos é na abstração Ambiente Virtual do Usuário, cujos componentes para implementação estão sendo construídos pelo EXEHDA (instanciação remota, base de dados e código *pervasiva*, perfil do usuário, contexto). Os trabalhos de pesquisa para exploração de todas as possíveis funcionalidades decorrentes do AVU estão em fase inicial no ISAM.

Observa-se, ao final dessa análise, que *middleware*s que atendam grande parte dos requisitos impostos pela *pervasividade* ainda não estão disponíveis. Somente soluções parciais que abordam aspectos específicos necessários à infra-estrutura de suporte à Computação Pervasiva em escala global estão em desenvolvimento. No quadro 2.2 está resumida esta situação, no mesmo os principais projetos abordados neste capítulo estão associados aos requisitos da Computação Pervasiva que atendem.

2.9 Necessidade de uma nova proposta de *middleware*

O projeto de aplicações pervasiva é dirigido por dois conjuntos de restrições. Primeiro, os recursos locais nos dispositivos móveis como peso, tamanho, consumo de energia, poder de processamento e capacidade de armazenamento, são limitados, em comparação com os sistemas fixos de custo similar. Segundo, a infra-estrutura de suporte é altamente variável (conectividade, recursos remotos), especialmente em comparação com o suporte mais estático utilizado nos sistemas distribuídos tradicionais. Neste ambiente, desenvolver aplicações móveis é difícil porque o projetista deve conhecer o domínio da aplicação e o da rede. Por essas razões, pesquisadores concordam que as aplicações móveis devem ser adaptativas: dependendo das características do ambiente de execução, aplicações alteram seu comportamento visível e/ou o modo como utilizam os escassos recursos. Linguagens, bibliotecas ou *middlewares* podem potencialmente auxiliar nesta tarefa, porém, nas aplicações que existem hoje, o processo de adaptação ao contexto é altamente específico de uma aplicação e, na maioria das vezes, *ad-hoc*. Esta realidade é um complicador real, quando o objetivo é o desenvolvimento de aplicações de propósito geral.

Existe, portanto, requisitos emergindo para uma nova classe de serviços de sistemas distribuídos projetados especificamente para este ambiente dinâmico. Tais serviços devem evitar suposições sobre o ambiente básico de suporte, os quais impedem-no de

operar eficientemente através da variedade de redes. Este nova classe de serviços é chamada de serviços adaptativos. O potencial para sua aplicabilidade é considerável, pois poderão explorar totalmente o nível de conectividade disponível em um dado momento, além de serem portáveis.

Considerando o propósito do EXEHDA de atender a perspectiva do ISAM de trabalhar com: (*i*) equipamentos cuja natureza abrange *palmtops*, *desktops* e *workstations* e, portanto, significativamente diferentes no tocante ao perfil computacional; (*ii*) suporte à mobilidade física e, assim, viabilizar uma possível troca de ambiente onde acontece o processamento; e (*iii*) um meio físico de execução de abrangência global, sujeito a elevadas flutuações na disponibilidade de recursos, se faz necessário que esta nova proposta de *middleware* expresse consciência de si próprio e do ambiente computacional através da capacidade de adaptar-se às condições dos recursos disponibilizados pelo meio físico de execução.

Devido ao fato de que os *middleware*s analisados não atendem adequadamente os requisitos de suporte à arquitetura ISAM, propôs-se um ambiente para execução de aplicações móveis, distribuídas, que contempla suporte à sensibilidade de contexto (*context-aware*), à semântica *siga-me*, e à uma estratégia colaborativa entre a aplicação e o *middleware* nas decisões de adaptação. Para manter o custo computacional da estratégia adaptativa minimizado, este *middleware* emprega uma estratégia reflexiva [YAM 2003, YAM 2002, YAM 2002a, YAM 2002b].

Aspecto	**Funcionalidade**	Aura	Gaia	Lime	Tota	C. Toolkit	Solar	Columba	Globus
Mobilidade	Suporte à mobilidade de dispositivos	■	■		■		■		
	Suporte à mobilidade de código	■	■	■	■		■	■	
	Suporte à mobilidade de usuário	■	■						■[1]

Quadro 2.2: Principais projetos e os requisitos da Computação Pervasiva (continua)

Programação de aplicações	Suporte à distribuição	■	■	■	■				■
	Suporte à adaptação dinâmica	■	■		■				
	Suporte à definição do contexto de interesse				■	■	■		
Ambiente Pervasivo	Suporte à heterogeneidade de dispositivo e de software básico	■	■	■	■	■	■	■	
	Suporte à monitoração	■[2]	■[2]			■	■		
	Suporte ao reconhecimento de contexto					■	■		
	Suporte ao acesso pervasivo de dados e código	■							■[3]
	Suporte à escalabilidade				■		■		■
	Suporte à pro-atividade	■	■						
	Suporte à adaptação de serviços do *middleware*	■							
	Suporte ao desacoplamento temporal e espacial			■	■				
	Suporte à descoberta de recursos						■		

1- restrita a possibilidade de autenticação em escala global (*single sign-on*)
2 - em contextos específicos
3 - precisa ser controlada explicitamente

Quadro 2.2: Principais projetos e os requisitos da Computação Pervasiva

No próximo capítulo é discutido como foram construídos os princípios que nortearam a concepção do EXEHDA, e qual o impacto destes princípios na definição dos seus requisitos de modelagem.

3 EXEHDA: PRINCÍPIOS GERAIS

The important thing is not to stop questioning.

- Albert Einstein

Este capítulo trata dos princípios gerais que nortearam a concepção do EXEHDA. Como já introduzido, o EXEHDA é um *middleware* direcionado às aplicações distribuídas, móveis e conscientes do contexto da Computação Pervasiva [YAM 2003a]. Em um sistema pervasivo tem-se a combinação da mobilidade lógica com a mobilidade física. Neste texto, entende-se por mobilidade lógica a reorganização da relação entre os componentes de software da aplicação; esta reorganização poderá ou não disparar uma migração de software. Por sua vez, fica caracterizada uma migração de software quando ocorrer uma troca no equipamento que hospeda o componente da aplicação. Por fim, por mobilidade física entende-se o deslocamento do usuário empregando, ou não, um dispositivo portátil.

3.1 Taxonomia das aplicações

Augustin, Yamin, Barbosa e Geyer, baseados nos trabalhos realizados no âmbito do Projeto ISAM, categorizaram as aplicações que contemplam mobilidade lógica e física, segundo os aspectos que influenciam sua adaptação. Esta categorização culminou em uma primeira taxonomia para este tipo de aplicação, a qual foi objeto de uma publicação internacional [AUG 2002].

A taxonomia proposta para aplicações adaptativas com mobilidade lógica e física está representada na figura 3.1. A classificação foi construída a partir da resposta a três questões: “quem executa a adaptação – aplicação ou o *middleware*?”, “quem aciona a adaptação – recursos, localização ou contexto?”, “como é feita a adaptação – paramétrica, conteúdo ou funcional?”. Na figura 3.1, as elipses representam os tipos de aplicações, as linhas horizontais (A, B e C) representam a sensibilidade ao elemento computacional, enquanto que as linhas verticais (X, Y e Z) representam as estratégias de adaptação.

Esta taxonomia foi conseqüência da necessidade de organizar conceitos relativos às áreas que integram a Computação Pervasiva, particularmente no que diz respeito à mobilidade física. O fato de a Computação Pervasiva ser bastante recente faz com que o escopo das suas necessidades não esteja ainda estabelecido, e, de forma análoga a outras áreas especializadas da Ciência da Computação, a respectiva teoria está sendo consolidada após uma prática inicial real, em meio a uma oferta de produtos e serviços e de uma inevitável controvérsia sobre conceitos e nomenclatura.

Na Computação Pervasiva, as aplicações devem ter a propriedade de adaptabilidade ao contexto [YAM 2002, DEY 2001, AUG 2001]. Pode-se dizer que a adaptação consiste de alterações funcionais ou não-funcionais, para atender requisitos oriundos de novos usos, ou novas condições do meio onde está se realizando o processamento. A noção de adaptação na Computação Pervasiva requer ainda definições de consenso. Observa-se que a adaptação nos sistemas móveis pode se referir a diversas noções, dependendo dos objetivos e domínio das aplicações. Por exemplo, em muitos sistemas, a adaptação diz respeito ao uso de técnicas de transformação dos dados para trafegarem na rede, sendo este processo disparado pela alteração na largura da banda.

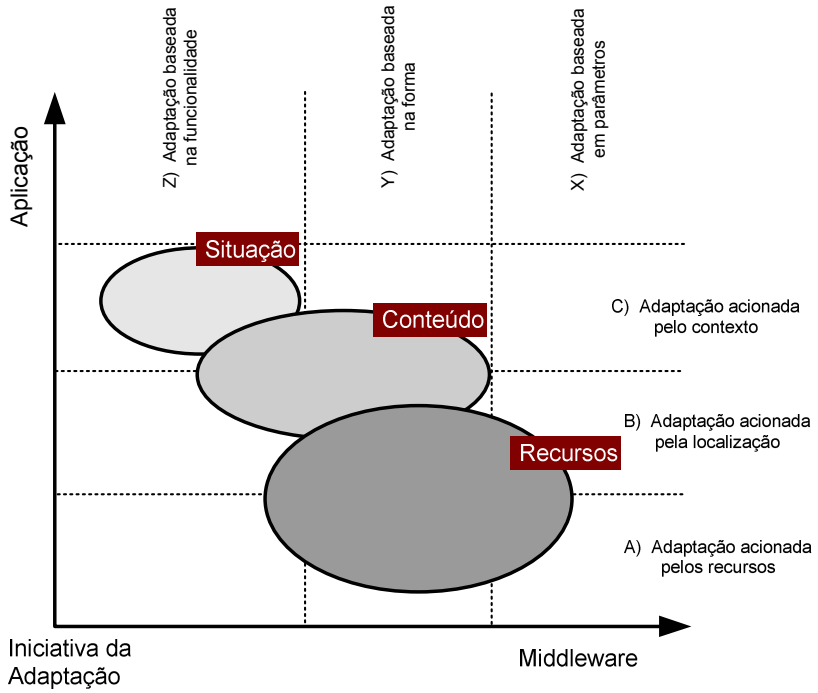


Figura 3.1: Taxonomia da adaptação de aplicações com mobilidade lógica e física

O pouco consenso na comunidade científica sobre como administrar a adaptação faz com que os diversos grupos de pesquisa postulem o uso de mecanismos adaptativos em diferentes níveis da arquitetura de software, desde a aplicação passando por diferentes níveis da infra-estrutura de suporte [AUG 2004, RAN 97].

Da mesma forma, o termo contexto pode referir-se a variadas noções [DEY 2001]. Neste trabalho, o contexto de execução é definido por elementos computacionais que podem ser mensurados e obtidos, tais como poder computacional e memória disponíveis no processador, banda-passante e latência do canal de comunicações, consumo de energia, localização e preferências do usuário.

3.1.1 Principais tipos de aplicações móveis

No quadro 3.1 estão resumidos os principais tipos de aplicações móveis identificadas a partir da análise da literatura. Este estudo foi central na definição dos componentes da arquitetura de software proposta neste trabalho (vide item 3.4), os quais procuram buscam respostas para as questões: por que, quem, com o quê, quando, onde e como prover um comportamento adaptativo ao contexto para estas aplicações no âmbito da Computação Pervasiva.

Tipo	**Descrição**	**Propriedade-Foco**
Aplicações **Nômades**	Processam em equipamentos móveis como PDAs e celulares (*SmartPhones*). Seu principal uso é direcionado à administração pessoal. Não costumam utilizar procedimentos adaptativos, porém as aplicações devem considerar o consumo de energia e os recursos computacionais restritos.	Portabilidade
Aplicações **conscientes da rede** (*network-aware*)	Adaptam-se aos recursos disponibilizados pela infra-estrutura de rede. Seu objetivo é a continuidade do serviço, apesar de perdas em componentes do resultado final. Seu principal contexto é a largura de banda, e usualmente são empregados procedimentos adaptativos baseados em filtragem/modificação dos dados para otimizar o fluxo dos mesmos através da rede. Com freqüência menor são empregados procedimentos de adaptação baseados em *caching* e/ou replicação [LOW 98, STE 99].	Portabilidade, conectividade e adaptabilidade a recursos da rede
Aplicações **conscientes dos recursos** *(resource-aware*)	As aplicações buscam identificar e empregar os recursos disponíveis. A indisponibilidade do recurso pode levar à busca de outra localização onde este possa estar. Tipicamente esta classe de aplicações emprega a tecnologia de agentes móveis [PHA 98]. A descoberta de recursos e a migração de software são características desejadas para este tipo de aplicação [NOB 2000, RAN 97].	Mobilidade, conectividade e adaptabilidade aos recursos locais
Aplicações **conscientes da localização** (*location-aware*)	Estas aplicações empregam a informação de localização do usuário como referência para refinar a execução de ações, bem como para fornecer informações sobre o que está próximo ao usuário. O reconhecimento da posição onde se encontra o usuário é central, e este tipo de aplicação pode ser considerado um subtipo das aplicações conscientes do contexto [WAN 2001].	Adaptabilidade à localização, mobilidade, portabilidade e conectividade

Quadro 3.1: Tipos de aplicações com mobilidade lógica e física (continua)

Aplicações **conscientes do contexto** (*context-aware*)	O contexto caracteriza informações sobre o estado dos dispositivos envolvidos no processamento, sobre os recursos de rede, sobre a localização do usuário, dentre outros. Deste modo, o monitoramento do contexto, a modelagem das informações provenientes do contexto e a notificação das mesmas são aspectos centrais para estas aplicações. Faz-se necessária à existência de configurações alternativas de acordo com o contexto, e deverá existir uma efetiva colaboração entre o sistema e a aplicação na gerência dos procedimentos adaptativos [CHE 2001, DEY 2000].	Mobilidade, portabilidade, conectividade e adaptabilidade ao contexto (lógico e físico)
Aplicações **conscientes da situação** (*situation-aware*)	Nos tipos anteriores de aplicação, a adaptação acontece a partir da aplicação, que se ajusta ao meio, com ou sem a colaboração do ambiente de execução. Nas aplicações conscientes da situação, a decisão de adaptação considera fatores externos à aplicação, e pode ser realizada pelo ambiente de execução considerando a existência de outras possíveis aplicações ao redor. Estas outras aplicações passam deste modo a constituir também elementos de contexto. É introduzido um caráter social na proposta de adaptação. Em função do nível mais elevado de abstração que caracteriza este tipo de aplicação, não existem, pelo investigado até agora, trabalhos na direção deste tipo de aplicação.	Mobilidade, conectividade, adaptabilidade ao contexto e pró-atividade do ambiente de execução

Quadro 3.1: Tipos de aplicações com mobilidade lógica e física

3.2 O perfil da aplicação-alvo

Hoje, as aplicações direcionadas para dispositivos de elevada portabilidade como PDAs, Smartphones e assemelhados, ainda são caracterizadas por um perfil nômade. O usuário opera em uma perspectiva onde os programas e os dados estão disponíveis localmente, no próprio equipamento. Neste caso, quando necessário e por iniciativa do usuário, é executada uma operação de sincronização com um equipamento externo, a qual é utilizada para envio e recebimento de programas e dados, e sobretudo para realização de cópias de segurança.

Apesar de bastante disseminado, o modelo nômade não atende à demanda de serviços introduzida com a atual dinamicidade da mobilidade física do usuário, portando ou não seu dispositivo móvel. Entende-se que os recursos empregados no atendimento ao usuário estão geograficamente distribuídos em uma rede de abrangência global. Nesta perspectiva, os componentes que integram o meio de execução, dentre estes, dados, códigos, equipamentos e serviços, deverão ser gerenciados por um ambiente de execução, que viabilize um acesso independente do equipamento utilizado, do local e do momento de acesso aos mesmos (acesso pervasivo).

Além disso, as aplicações são disponibilizadas segundo a semântica *siga-me*, a qual define que o usuário executa suas aplicações no local em que se encontra, mesmo em deslocamento. Para isto, neste ambiente de execução, os usuários dispõem de um ambiente virtual, que poderá ser acessado independentemente da localização onde este se encontrar, ou do dispositivo que dispuser no momento, e o código das aplicações é enviado sob demanda. Logo, este código deve estar adaptado ao contexto corrente de uso da aplicação.

Desta forma, as aplicações são distribuídas, tem mobilidade lógica e física e precisam ser adaptativas ao contexto em que serão processadas.

Os equipamentos portáteis não têm a incumbência de armazenar código e dados de forma persistente, mas sobretudo atuam como portais do poder computacional distribuído na rede global. As aplicações são influenciadas por políticas definidas na etapa de desenvolvimento, as quais orientam o ambiente de execução durante a tomada de decisão para seu processamento.

Este perfil comportamental foi concebido com uma divisão de responsabilidades entre quem desenvolve a aplicação e o ambiente de execução, implementado como um *middleware*. Daí decorre a total integração dos esforços de pesquisa do EXEHDA, com os pertinentes ao ISAMadapt – ambiente de desenvolvimento de aplicações [AUG 2004].

3.3 O modelo de contexto para aplicação-alvo

O deslocamento do usuário, acompanhado ou não do seu equipamento, gera um cenário complexo no que diz respeito ao contexto. A taxonomia de aplicações apresentada na seção 3.1 é uma indicadora modesta desta situação. O usuário, em função da sua localização, fica à mercê de recursos e serviços, cuja efetiva disponibilidade oscila também no tempo. Esta oscilação é devida, dentre outros, a aspectos como:

- a heterogeneidade de rede/recursos;
- a troca no local de concentração dos usuários ou do processamento introduzida pela mobilidade lógica e física.

A figura 3.2, a seguir, sintetiza esta visão.

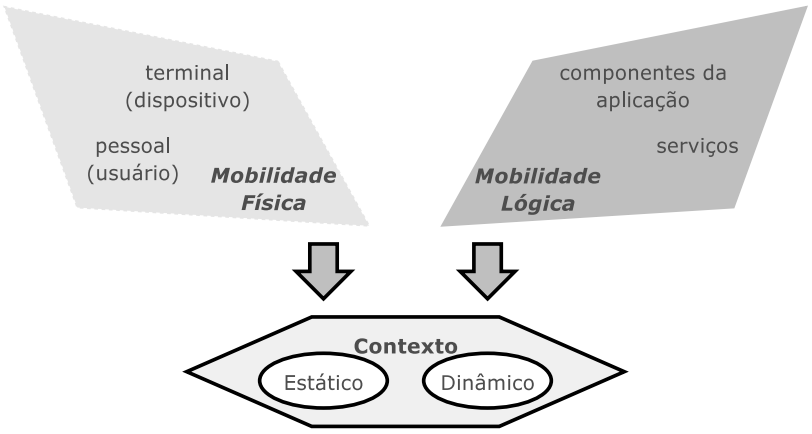


Figura 3.2: Aspectos do contexto na Computação Pervasiva

As diferentes naturezas das aplicações necessitam de abstrações particulares do contexto. Neste sentido, a sua programação deve priorizar os elementos do contexto que são relevantes para o seu escopo de interesse, e ignorar outros.

O modelo de contexto adotado pelo ISAM foi concebido no desenrolar das pesquisas do EXEHDA e do ISAMadapt, e de forma análoga aos outros trabalhos na área, tem uma concepção informal [CHE 2001, DEY 2000]. Esta concepção informal, até então usual entre os trabalhos relacionados a *context-aware applications*, é decorrente sobretudo das atividades de pesquisa na área estarem no seu início, mesmo no nível internacional.

A continuidade dos trabalhos, e o conseqüente aumento no volume de aplicações desenvolvidas, irão trazer subsídios para os esforços na busca de modelos formais para expressar o contexto [HEN 2002]. A seguir será resumido o modelo de contexto adotado como referência para o EXEHDA, e todos os outros projetos relacionados ao ISAM.

3.3.1 Modelo de contexto

No ISAM, o contexto é definido como “toda informação relevante para a aplicação e que pode ser obtida por esta”. O programador explicitamente identifica os aspectos da entidade de onde provém a informação e define seus atributos (elementos de contexto), os quais passam a integrar o contexto da aplicação [ISA 2002]. Por exemplo, um nodo de processamento poderá ter como elemento de contexto: a carga computacional, a ocupação de memória, o tamanho da fila de processos, etc. A alteração em um destes atributos poderá ser utilizada para disparar um procedimento de adaptação, tanto na aplicação como no próprio ambiente de execução.

Pode-se definir um modelo de contexto associando-o a “um conjunto de atributos que descrevem o estado das entidades”. Entidades são elementos abstratos do modelo e representam a fonte da informação. Atributos são associados a objetos específicos – entidades – e são chamados de elementos de contexto. O conjunto dos elementos de contexto de interesse da aplicação definirá seu contexto. O relacionamento entre os elementos de contexto e a entidade é estabelecido explicitamente por relações que definem como é a coleta, qual a origem e o tipo da informação. O estado do elemento de contexto é uma das possibilidades previstas à qual a aplicação pode se ajustar. O estado é monitorado e o dado coletado é interpretado para traduzir uma possibilidade válida, gerando um dado contextualizado.

O modelo de contexto para cada aplicação é a descrição da dependência entre o estado do contexto, e sua interpretação tanto pela aplicação, como pelo *middleware* que gerencia sua execução (visão particular).

No ISAM, para representação do modelo de contexto é adotada uma notação baseada em objetos, acrescida de outra específica para caracterização dos relacionamentos determinados pelo contexto. Esta notação está apresentada na legenda da figura 3.3. Esta figura sintetiza o contexto de uma aplicação, na qual o estado do contexto é modelado como um atributo da entidade da qual é obtido. A indicação de como o contexto é gerado na visão de uma aplicação em particular, é expresso por arcos que ligam a entidade ao elemento de contexto. A complexidade do *middleware* para fornecer informação de contexto advém das estruturas conceituais, que devem ser amplas o suficiente para tratar com diferenciados tipos de contexto, deste modo

sofisticadas o bastante para fazer as distinções necessárias para as aplicações, e por outro lado simples para fornecer uma base prática para a programação.

As informações coletadas, e que caracterizam o contexto em que está ocorrendo a execução, exibem aspectos diversos e são passíveis de diferentes caracterizações [CHE 2001]. Na próxima seção estão classificados os principais aspectos das informações de contexto, de interesse do projeto ISAM e, particularmente, do EXEHDA.

3.3.2 Classificação das informações de contexto

A classificação adotada aborda três aspectos: temporal, uso e tratamento da informação.

Da temporalidade da informação:

- **Informações estáticas**: descrevem aspectos que não se alteram com o tempo, por exemplo, atributos relativos ao tipo de equipamento. As características estáticas são obtidas a partir de perfis construídos de forma automática ou pelo usuário. Estes perfis ficam registrados através de arquivos descritores de configuração;
- **Informações dinâmicas:** traduzem aspectos do contexto que oscilam com freqüência, por exemplo a ocupação do processador e a localização do usuário. Este tipo de informação é obtido através de um serviço de monitoramento, que atua periodicamente ou ativado por eventos. As informações monitoradas podem ficar imprecisas por diversos motivos; dentre estes se destacam: atrasos de propagação através da rede, desde o momento da geração até o uso da informação monitorada, falha no sensor ou no algoritmo de tratamento, perdas de conexão com o sensor ou com o usuário da informação monitorada, etc.

O tema monitoramento dos aspectos tanto de software como de hardware vem sendo tratado por diversos grupos [FOX 98, MIL 2000]. Particularmente, no projeto ISAM têm-se os trabalhos [SIL 2001, SIL 2003, ARA 2001, ARA 2002].

Do uso da informação de estado do contexto:

- **Direto:** quando a informação monitorada pode ser utilizada de forma bruta como informação de contexto. Via de regra, traduz uma informação corrente – atual – do meio monitorado, por exemplo a ocupação do processador;
- **Interpretado:** neste caso a informação monitorada é processada antes de ser utilizada, por exemplo a localização do usuário: casa, escritório, etc.

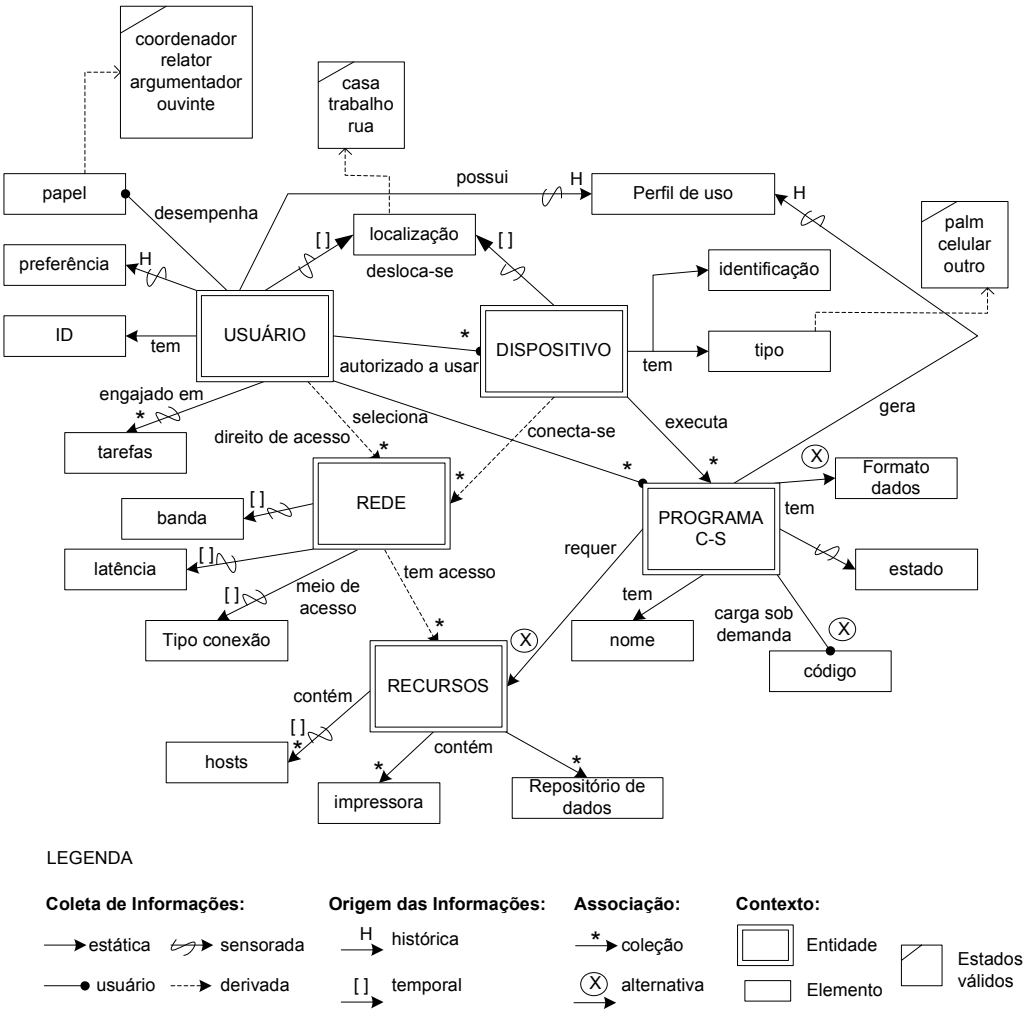


Figura 3.3: Modelo de contexto

Do tratamento da informação obtida:

- **Corrente**: neste caso a informação monitorada traduz um evento atual, podendo ser alvo de processamento como filtragem e/ou refinamento. É importante observar que as diversas aplicações podem requerer diferentes interpretações para um mesmo dado monitorado;
- **Histórica**: as informações monitoradas neste caso constituem séries históricas com o objetivo de prever o futuro. Estas séries são constituídas por registros persistentes dos dados monitorados;
- **Derivada**: as informações de contexto ainda podem ser formadas pela composição de informações mais simples. Um exemplo neste sentido diz respeito à localização; este tipo de informação pode indicar a atividade provável do usuário ou que estabelecimentos estão próximos a ele. Esta situação traduz a existência de relacionamentos entre elementos do contexto, os quais podem ser

considerados para aumentar a certeza quando da interpretação do contexto.

A habilidade de se ajustar às alterações no contexto do meio de execução é uma característica básica do software destinado à Computação Pervasiva. Adaptar-se em função do meio, exige consciência do contexto. A seguir, ver-se-á como isto é refletido na concepção da arquitetura ISAM.

3.4 A arquitetura de software

A figura 3.4 apresenta uma visão geral da arquitetura de software para o ISAM. A representação da consciência do contexto nesta figura como um módulo virtual tem por objetivo ressaltar sua importância na arquitetura, bem como caracterizar sua presença na concepção de todos os outros componentes.

Com o intuito de minimizar o custo de especificar os aspectos necessários para o tratamento da semântica *siga-me* relacionados à mobilidade lógica e física, de distribuição e de adaptação, os mecanismos referentes a tais aspectos oferecidos na linguagem ISAMadapt estão integrados ao ambiente de execução. Desta forma, a arquitetura ISAM, modelada com esta perspectiva, apresenta uma organização lógica em três camadas: (sup) camada de aplicação; (interm) camada de suporte e ambiente de execução; (inf) camada de sistemas básicos.

Na camada superior (aplicação) está a linguagem de programação (ISAMadapt), a qual disponibiliza abstrações para programação de aplicações distribuídas, móveis e conscientes do contexto direcionadas à Computação Pervasiva [AUG 2004, AUG 2002a, AUG 2002b]. A linguagem ISAMadapt emprega alguns conceitos derivados do Holoparadigma [BAR 2002].

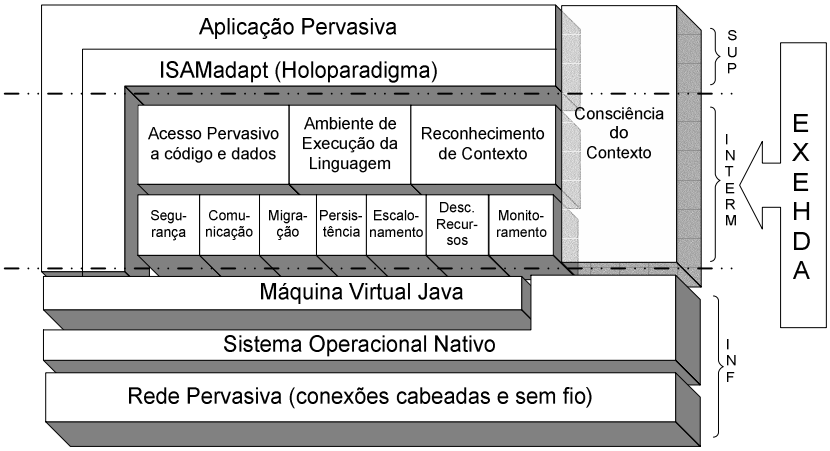


Figura 3.4: Visão Geral da Arquitetura ISAM

Na camada intermediária (EXEHDA) estão os mecanismos de suporte à execução da aplicação pervasiva e às estratégias de adaptação [YAM 2003, YAM 2003a, YAM 2002a, YAM 2002b, SIL 2003, REA 2004]. Esta camada é formada por dois níveis:

O primeiro nível é composto por três módulos de serviço à aplicação: Acesso Pervasivo a Código e Dados, Reconhecimento de Contexto e Ambiente de Execução da Aplicação.

- O acesso pervasivo a dados e código compreende os componentes que disponibilizam o Ambiente Pervasivo (ISAMpe), incluindo Ambiente Virtual do Usuário (AVU), Ambiente Virtual da Aplicação (AVA) e Base de Dados pervasiva das Aplicações (BDA). O Ambiente Virtual do Usuário compõe-se dos elementos que integram a interface de interação do usuário com o sistema. Este módulo é responsável por implementar o suporte para que a aplicação que o usuário está executando em uma localização possa ser instanciada e continuada em outra localização sem descontinuidade, permitindo o estilo de aplicações *siga-me* (follow-me) num ambiente pervasivo. O desafio da adaptabilidade para implementar o AVU é suportar os usuários em diferentes localizações, com diferentes sistemas de interação, que demandam diferentes sistemas de apresentação, dentro dos limites da mobilidade. Por sua vez, entende-se como AVA o conjunto de atributos que identifica uma execução específica de uma aplicação, enquanto a BDA constitui o repositório de códigos das aplicações em geral;
- O Ambiente de Execução da Linguagem é o encarregado pelo gerenciamento da aplicação durante seu tempo de vida;
- O Serviço de Reconhecimento de Contexto é responsável por informar o estado dos elementos de contexto de interesse da aplicação e do próprio ambiente de execução.

No segundo nível da camada intermediária estão os serviços básicos do EXEHDA, os quais provêm as funcionalidades necessárias para o primeiro nível e cobrem vários aspectos, tais como migração – mecanismos para deslocar um componente de software de uma localização física (equipamento) para outra; persistência – mecanismo para aumentar a disponibilidade e o desempenho do acesso aos dados; descoberta de recursos – para dar suporte ao movimento dos dispositivos móveis e dos componentes entre diferentes células; comunicação – com possibilidade de ser anônima e assíncrona; escalonamento – permite decidir o melhor nodo para criar os componentes da aplicação; monitoramento – sensores que fornecem informações sobre o ambiente de execução e aplicação.

A camada inferior da arquitetura é composta pelos sistemas e linguagens nativas que integram o meio físico de execução. Por questões de portabilidade, nesta camada a plataforma base de implementação é a Máquina Virtual Java em suas diferentes abordagens. Duas plataformas são utilizadas prioritariamente: (*i*) J2SE (Java Standard Edition, <http://java.sun.com/j2se/>), e (*ii*) J2ME (Java Micro Edition, <http://java.sun.com/j2me/>). A arquitetura supõe a existência de uma rede global com suporte à operação sem fio, interligada a outra cabeada que disponibilize uma infra-estrutura de equipamentos e serviços em escala global (rede pervasiva).

Como visto, a arquitetura ISAM está organizada em camadas lógicas, com níveis diferenciados de abstração, e está direcionada para a busca da manutenibilidade da qualidade de serviços oferecida ao usuário móvel através do conceito de adaptação. Nesta, o sistema se adapta para fornecer qualidade dos serviços prestados, enquanto que

a aplicação se adapta para atender a expectativa do usuário móvel, mantendo a funcionalidade da aplicação.

3.5 O controle da adaptação

Para o EXEHDA a adaptação não é uma propriedade funcional da aplicação. Assim seu tratamento pode acontecer de forma autônoma, realizado pela gerência da execução da aplicação. Durante o projeto de um sistema adaptável, seja no nível de aplicação seja no de *middleware*, é necessário identificar e qualificar as características que representam aspectos a serem considerados, de modo que decisões de adaptabilidade possam ser tomadas. Considera-se que a adaptação ocorrerá se o sistema dispuser dos recursos necessários para sua efetivação, descartando-se situações onde o nível de disponibilidade de recursos inviabiliza o processamento.

3.5.1 Etapas da adaptação

A modelagem do processo de adaptação, esquematizada na figura 3.5, foi dividida em três etapas:

1. **detecção de alterações**, que engloba monitoração, interpretação e notificação, visa observar o ambiente para detectar e notificar alterações significativas. Esta pode ser realizada de duas formas: *pull*, a informação é solicitada por uma requisição; *push*, a informação é enviada ao cliente que se inscreveu no serviço;
2. **escolha da ação**, engloba a seleção da ação adaptativa entre alternativas pré-definidas pelo programador. A seleção pode ser realizada periodicamente, ou quando ocorre o evento de notificação de alteração;
3. **ativação da ação**, refere-se à execução da ação adaptativa selecionada.

A adaptação se refere à alteração no comportamento, na estrutura ou na interface da aplicação, em resposta a trocas arbitrárias no estado do elemento de contexto. Os tipos de adaptações que podem ser empregadas pela aplicação dependem da natureza desta e dos recursos que ela requer. Alguns exemplos de comportamento adaptativo incluem: (*i*) variedade de filtros (compressão, omissão, conversão de formato) inseridos entre o servidor e o cliente; (*ii*) alteração da fidelidade de saída; (*iii*) interface reduzida; (*iv*) migração de componentes para máquinas mais poderosas; (*v*) funcionalidade conectada em face à desconexão utilizando *buffering* e *prefetching*.

Considera-se que as abstrações dos sistemas operacionais/linguagens de programação existentes não são suficientes, nem apropriadas para tratar as necessidades das aplicações pervasivas. No projeto ISAM, defende-se um estilo de programação que oferece ao desenvolvedor a opção de que a aplicação exiba explicitamente um comportamento adaptativo que será gerenciado e executado por um ambiente de execução (*middleware*). O código da aplicação é concebido de forma adaptativa, e reflete as possibilidades de comportamento da aplicação face à alteração no estado do elemento do contexto ao qual a aplicação é sensível [AUG 2004].

3.5.2 Componentes para expressividade do contexto

A análise feita para concepção da taxonomia apresentada na seção 3.1, com suas variadas noções de adaptação, permitiu identificar os requisitos de propósito geral para expressar adaptabilidade ao contexto, os quais são:

- descrição do elemento de contexto de interesse: identificação dos elementos que compõem o ambiente e que modelam uma visão particular de contexto;
- descrição do comportamento adaptativo: coleções de ações e o respectivo conjunto de restrições onde estas podem ocorrer. Adaptação pode incluir alterações internas (parâmetros, algoritmos ou representação de dados) ou alterações externas (reconfiguração, como migração de componentes);
- descrição de políticas: regras de propósito geral ou particular que orientam as decisões de adaptação realizadas pelo *middleware*.

3.5.3 A adaptação colaborativa multinível

A adaptação deve ser dinâmica, devido às flutuações no estado do contexto, as quais ficam potencializadas quando do deslocamento do usuário, e/ou dos componentes da aplicação; deve ser automática, quando possível; e gerenciada de forma colaborativa com a aplicação. Diferentemente dos outros sistemas que contemplam transparência na gerência da mobilidade tanto lógica como física, para o EXEHDA a aplicação deve ser *adaptation-aware*. Nesta ótica, as aplicações podem participar do processo de adaptação, se assim o programador desejar (vide figura 3.5).

Deste modo, para atender o requisito de *adaptation-aware*, foi proposto para a arquitetura de software ISAM um modelo denominado Adaptação Colaborativa Multinível [YAM 2002b, YAM 2001], onde:

1- o *middleware* é responsável por adaptações, normalmente relacionadas ao desempenho e à gerência de recursos e serviços;

2- a aplicação é responsável por decisões de adaptações específicas de seu domínio e situações de uso;

3- ambos são responsáveis por uma negociação de decisões de adaptação.

Muitas abordagens enfatizam a transparência do gerenciamento dos aspectos não-funcionais como adaptação. Considera-se, no entanto, que esta abordagem “caixa-preta” não é apropriada para o desenvolvimento de aplicações pervasivas com o grau de flexibilidade e generalidade perseguido pelo projeto ISAM. Desta forma, a proposta para minimizar o conflito entre transparência e consciência dos aspectos da adaptação é a colaboração entre aplicação e *middleware* no gerenciamento da adaptação.

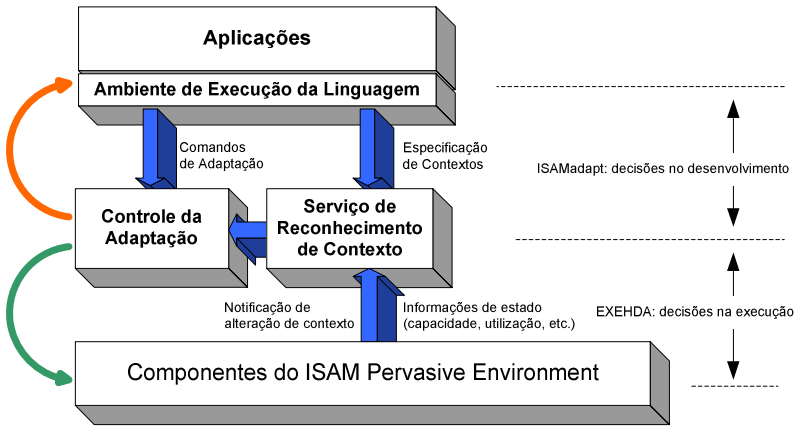


Figura 3.5: Adaptação colaborativa multinível

Na perspectiva do EXEHDA, além de cooperar na adaptação da aplicação, o *middleware* também se adapta ao contexto, daí a adaptação ser colaborativa e multinível [YAM 2002b]. Os principais elementos de contexto hoje considerados no EXEHDA para sua adaptação são: o tipo do equipamento, o estado de ocupação dos seus recursos e a situação de sua conectividade no momento.

3.6 Identificando requisitos de um *middleware* para a arquitetura ISAM

Considerando as motivações para este trabalho, considerando o estudo dos aspectos introduzidos pela área da Computação Pervasiva, associado à análise de trabalhos relacionados com o EXEHDA, e por fim tendo como referência o escopo de atuação do ISAM, nesta seção estão identificados os requisitos que deverão ser atendidos pelo EXEHDA.

Além dos requisitos que o *middleware* deve atender para viabilizar a execução de aplicações na Computação Pervasiva, também estão relacionados os requisitos a serem contemplados quando da sua própria concepção.

A figura 3.6 resume as tecnologias que o EXEHDA se propõe a integrar, ficando caracterizada a adaptação ao contexto como um aspecto central para sua efetiva convergência.

Nesta seção, os requisitos estão elencados, sendo os mesmos analisados quando da descrição dos serviços e procedimentos necessários para a sua consecução (vide capítulo 4).

3.6.1 Do ponto de vista da aplicação

Considerando o perfil da aplicação-alvo (vide seção 3.2), e as características correspondentes, o *middleware* para suportá-la deverá atender os seguintes requisitos:

- suporte à execução distribuída;
- comunicação com desacoplamento espacial e temporal;
- acesso *pervasivo* a dados e a código;
- suporte à mobilidade lógica;

- suporte à mobilidade física;
- fornecimento de informações de contexto;
- suporte à adaptação dinâmica de aspectos funcionais;
- suporte à adaptação dinâmica de aspectos não-funcionais;
- política cooperativa com a aplicação nas decisões de adaptação.

3.6.2 Do ponto de vista do *middleware*

No tocante à construção do próprio *middleware*, os seguintes requisitos são identificados como necessários para atender a arquitetura de software da proposta ISAM (vide seção 3.4):

- **Adaptação ao contexto**: prevista para ocorrer durante a carga dos diferentes serviços. Esta adaptação é direcionada pelo recurso em que o serviço vai ser carregado, e tem por objetivo atender a situação de elevada heterogeneidade de recursos na Computação Pervasiva;
- **Economia de recursos**: o número de serviços que o *middleware* disponibiliza pode ser elevado, podendo estar em constante crescimento e/ou atualização. Além disso, dispositivos do ambiente pervasivo podem oferecer diferentes níveis de recursos. A perspectiva é instalar o mínimo necessário para que os nodos possam operar;
- **Operação desconectada**: durante os períodos em que um equipamento estiver no modo de desconexão planejada, o *middleware* precisa se manter consistente e operacional.

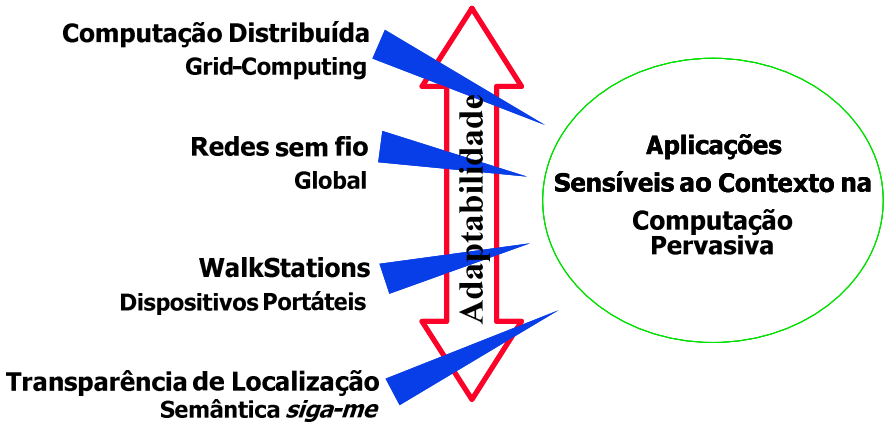


Figura 3.6: Tecnologias envolvidas no EXEHDA

Resumindo, tem-se que os principais requisitos que o EXEHDA deve atender são: (*i*) gerenciar tanto aspectos não-funcionais como funcionais da aplicação, e de modo

independente, (*ii*) dar suporte à adaptação dinâmica de aplicações; (*iii*) disponibilizar mecanismos para obter e tratar informações de contexto; *(iv)* utilizar informações de contexto na tomada de decisões, (*iv*) decidir as ações adaptativas de forma colaborativa com a aplicação e (*v*) disponibilizar a semântica *siga-me*, facultando ao usuário o disparo de aplicações e o acesso a dados a partir de qualquer lugar, e a execução contínua da aplicação em face ao seu deslocamento. Os princípios gerais do EXEHDA foram publicados em [YAM 2001] e [YAM 2002]. Por sua vez os aspectos relacionados ao comportamento adaptativo foram discutidos em [YAM 2002a] e [YAM 2002b].

O próximo capítulo detalha como estes requisitos foram considerados na modelagem do EXEHDA, sendo caracterizados os principais componentes e serviços disponibilizados.

4 EXEHDA: ASPECTOS DE MODELAGEM E DE SERVIÇOS

A journey of a thousand miles must begin with a single step

- Lao-tzu

No capítulo 3 foram construídos os requisitos a serem satisfeitos quando da proposta de um novo *middleware* destinado a atender as necessidades introduzidas pela Computação Pervasiva. Tais requisitos fundamentaram a proposição do EXEHDA, influindo na determinação do seu modelo, na organização de seus serviços e no conjunto de funcionalidades a serem disponibilizadas.

Neste capítulo, além dos aspectos de modelagem, será feita uma (*i*) descrição dos principais serviços que compõem o EXEHDA, (*ii*) da sua interdependência, (*iii*) e da sua contribuição na construção do suporte às premissas da Computação Pervasiva. Observe-se que a caracterização dos serviços está focada na descrição das funcionalidades providas através da sua interface, não sendo priorizados aspectos de implementação. Particularmente, a interface de cada serviço é ilustrada pela utilização da notação UML de diagramas de classe.

Entende-se por ISAMpe (ISAM *pervasive environment*) o ambiente computacional onde recursos e serviços são gerenciados pelo EXEHDA na perspectiva de atender os requisitos impostos pelo perfil da aplicação-alvo do projeto ISAM [ISA 2002]. Sua composição acontece tanto pelos dispositivos dos usuários, como pelos equipamentos da infra-estrutura de suporte, todos instanciados pelo seu respectivo perfil de execução do *middleware*.

O EXEHDA pode ser visto a partir de duas grandes perspectivas (vide figura 4.1):

1. **do ponto de vista das aplicações da Computação Pervasiva**: o EXEHDA é o provedor dos serviços que dão suporte às abstrações definidas quando do desenvolvimento. A interação das aplicações com o meio físico distribuído, através dos serviços disponibilizados pelo EXEHDA proporciona a estas aplicações a visão do ambiente pervasivo ISAMpe;
2. **do ponto de vista dos recursos que compõem o meio físico distribuído**: o EXEHDA define as políticas que normatizam a organização dos recursos e os mecanismos para sua gerência.

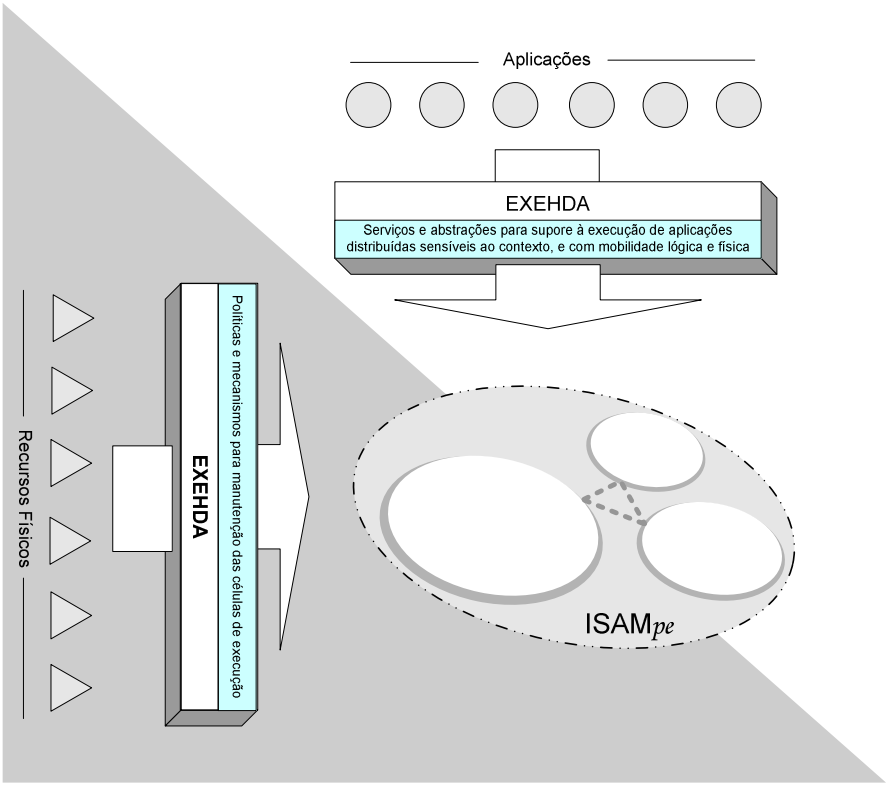


Figura 4.1: Visões de atuação do EXEHDA

Na seção 4.1 está caracterizada a organização do ISAMpe, e na continuidade deste capítulo é discutida a modelagem dos serviços do EXEHDA.

4.1 A organização do ISAMpe

A premissa de integrar os cenários (*i*) da Computação em Grade, (*ii*) da computação móvel e (*iii*) da computação sensível ao contexto, é mapeada em uma organização composta pela agregação de células de execução - EXEHDAcels, apresentada na figura 4.2.

O meio físico sobre o qual o ISAMpe é definido constitui-se por uma rede infra-estruturada, cuja composição final pode ser alterada pela agregação dinâmica de nodos móveis. Segundo a classificação construída no apêndice 2, este meio pode ser caracterizado como um sistema distribuído híbrido [YAM 2003].

Os recursos da infra-estrutura física são mapeados para três abstrações básicas, as quais são utilizadas na composição do ISAMpe:

- **EXEHDAcel**: denota a área de atuação de uma EXEHDAbase, e é composta por esta e por EXEHDAnodos. Os principais aspectos considerados na definição da abrangência de uma célula são: o escopo institucional, a proximidade geográfica e o custo de comunicação;
- **EXEHDAbase**: é o ponto de contato para os EXEHDAnodos. É responsável por todos os serviços básicos do ISAMpe e, embora constitua uma referência lógica única, seus serviços, sobretudo por aspectos de escalabilidade, poderão estar distribuídos entre vários

equipamentos. Isto é representado na figura 4.2 pelo sombreado associado ao símbolo da EXEHDAbase;

- **EXEHDAnodo**: são os equipamentos de processamento disponíveis no ISAMpe, sendo responsáveis pela execução das aplicações. Um subcaso deste tipo de recurso é o **EXEHDAnodo móvel**. São os nodos do sistema com elevada portabilidade, tipicamente dotados de interface de rede para operação sem fio e, neste caso, integram a célula a qual seu ponto-de-acesso está subordinado. São funcionalmente análogos aos EXEHDAnodos, porém eventualmente com uma capacidade mais restrita (por exemplo, PDAs).

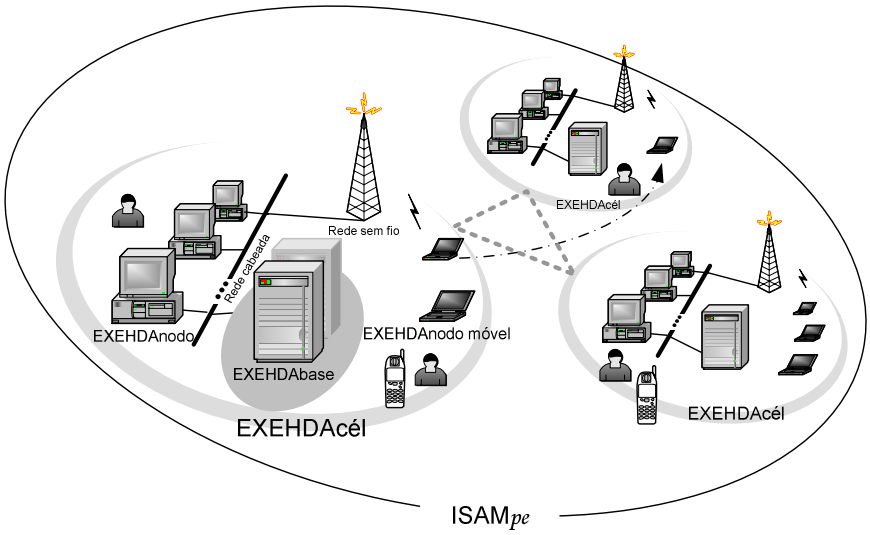


Figura 4.2: ISAM *Pervasive Environment*

O ISAMpe é formado por equipamentos multi-institucionais, o que conduz a um procedimento de gerência análogo ao praticado nos ambientes de Grade Computacional (*Grid Computing*) [GEY 2002, YAM 2003]. A forma como a organização celular do ISAMpe é gerenciada tem por objetivo central resguardar a autonomia das instituições envolvidas. A busca de soluções para o gerenciamento multi-institucional tem feito parte da agenda de pesquisa de projetos de Computação em Grade [FOS 2001] (vide seção 6.1 – Gerenciando o ISAMpe).

Apesar de não contemplar mecanismos de gerência específicos para recursos especializados como impressoras, scanners, etc., o EXEHDA faculta a catalogação de tais recursos como integrantes de uma determinada célula do ISAMpe, tornando-os, desta forma, passíveis de serem localizados dinamicamente, e, deste modo, poderem ser utilizados pelas aplicações pervasivas.

4.2 Suporte a semântica *siga-me*

Oferecer suporte à semântica *siga-me* é uma das contribuições centrais do EXEHDA ao Projeto ISAM. No EXEHDA, este suporte é construído pela agregação de funcionalidades relativas ao reconhecimento de contexto, ao acesso pervasivo e à comunicação [YAM 2004].

Como estratégia para tratamento da complexidade associada ao suporte da semântica *siga-me*, no EXEHDA é adotada a decomposição das funcionalidades de mais alto nível, recursivamente, em funcionalidades mais básicas, conforme ilustrado na figura 4.3.

Nesta perspectiva, no EXEHDA o reconhecimento de contexto está relacionado com dois mecanismos: (*i*) um de monitoração que permite inferir sobre o estado atual dos recursos e das aplicações, e (*ii*) outro que pode promover adaptações funcionais e não funcionais, tendo em vista o contexto monitorado.

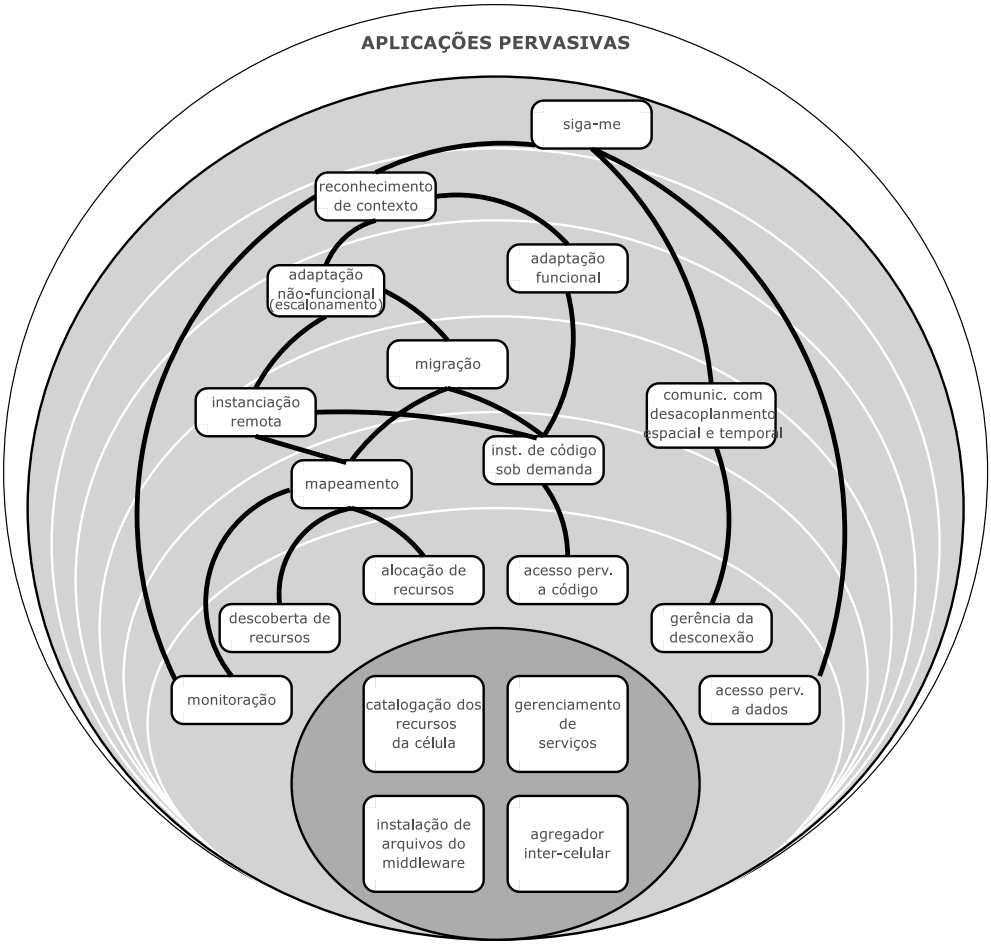


Figura 4.3: A semântica *siga-me* no EXEHDA

A adaptação não-funcional consiste na capacidade do sistema atuar sobre a localização física dos componentes das aplicações, seja no momento de uma instanciação do componente, seja, posteriormente, via migração do mesmo. Ambas operações demandam a existência de mecanismo para instalação sob demanda do código, assim como mecanismos para descoberta e alocação dinâmicas de recursos e acompanhamento de seu estado.

Por sua vez, a adaptação funcional consiste na capacidade do sistema atuar sobre a seleção da implementação do componente a ser utilizado em um determinado contexto de execução. Novamente surge a necessidade do suporte à instalação de código sob demanda.

A funcionalidade da instalação sob demanda implica que o código a ser instalado esteja disponível em todos os dispositivos nos quais este venha a ser necessário. Considerando as dimensões do ambiente pervasivo, é impraticável manter a cópia de todos os possíveis códigos em todos os eventuais dispositivos. Procede daí a necessidade de um mecanismo que disponibilize acesso pervasivo ao repositório de código, mecanismo este, que deve considerar fortemente o aspecto escalabilidade.

O aspecto de mobilidade, tanto dos componentes das aplicações quanto do usuário, inerente à semântica *siga-me*, faz propícia uma estratégia de comunicação caracterizada pelo desacoplamento espacial e temporal.

Outra característica da Computação Pervasiva é o usuário ter acesso ao seu ambiente computacional, independentemente de sua localização e do dispositivo empregado. Assim, de forma análoga ao repositório de código, é necessário um mecanismo que habilite o acesso pervasivo do usuário ao seu ambiente computacional, isto é, seus dados e configurações pessoais.

4.3 EXEHDA: organização baseada em serviços

O requisito de operação em um ambiente altamente heterogêneo, onde não só o hardware exibe capacidades variadas de processamento e memória, mas também as bibliotecas de software disponíveis em cada dispositivo, motivaram a adoção de uma abordagem na qual um núcleo mínimo do *middleware* tem suas funcionalidades estendidas por serviços carregados sob demanda. Esta organização reflete um padrão de projeto referenciado na literatura como *micro-kernel* [BUS 96]. Some-se a isto o fato de que esta carga sob demanda tem perfil adaptativo. Deste modo, poderá ser utilizada versão de um determinado serviço, melhor sintonizada às características do dispositivo em questão. Isto é possível porque, na modelagem do EXEHDA, os serviços estão definidos por sua interface, e não pela sua implementação propriamente dita.

A contra-proposta à estratégia *micro-kernel* de um único binário monolítico, cujas funcionalidades cobrissem todas as combinações de necessidades das aplicações e dispositivos, se mostra impraticável na Computação Pervasiva, cujo ambiente computacional apresenta elevada heterogeneidade de recursos de processamento conforme discutido na seção 1.2.1.

Por sua vez, o requisito do *middleware* de manter-se operacional durante os períodos de desconexão planejada motivou, além da concepção de primitivas de comunicação adequadas a esta situação, a separação dos serviços que implementam operações de natureza distribuída em instâncias locais ao EXEHDAnodo (instância nodal), e instâncias locais a EXEHDAbase (instância celular). Neste sentido, o relacionamento entre instância de nodo e celular assemelha-se à estratégia de Proxies, enquanto que o relacionamento entre instâncias celulares assume um caráter P2P. A abordagem P2P nas operações inter-celulares vai ao encontro do requisito de escalabilidade.

Com isso, os componentes da aplicação em execução em determinado dispositivo podem permanecer operacionais, desde que, para satisfação de uma dada requisição pelo *middleware*, o acesso a um recurso externo ao dispositivo seja prescindível. Por outro lado, a instância celular, em execução na base da célula, provê uma referência para os

outros recursos, no caso da realização de operações que requeiram coordenação distribuída. Neste sentido, observe-se que a EXEHDAbase é, por definição, uma entidade estável dentro da EXEHDAcel, permitindo que os demais integrantes (recursos) da célula tenham um caráter mais dinâmico no que se refere a sua disponibilidade (presença efetiva) na célula.

A organização do núcleo mínimo do EXEHDA**,** e o conjunto atual de serviços projetados para atender as necessidades da arquitetura ISAM, são apresentados nas seções seguintes. Os serviços do EXEHDA estão organizados em quatro grandes subsistemas: execução distribuída, adaptação, comunicação e acesso pervasivo (figura 4.4).

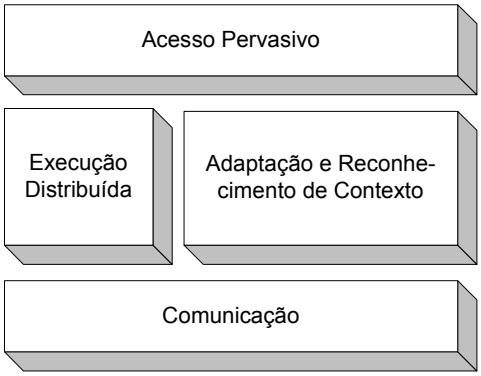


Figura 4.4: Organização dos subsistemas do EXEHDA

4.4 O núcleo do EXEHDA

A funcionalidade provida pelo EXEHDA é personalizável no nível de nodo, sendo determinada pelo conjunto de serviços ativos e controlada por meio de perfis de execução. Um perfil de execução define um conjunto de serviços a ser ativado em um EXEHDAnodo, associando a cada serviço uma implementação específica dentre as disponíveis, bem como definindo parâmetros para sua execução (seção 4.4.1).

Adicionalmente, o perfil de execução também controla a política de carga a ser utilizada para um determinado serviço, a qual se traduz em duas opções:

- quando da ativação do nodo (*bootstrap* do *middleware*);
- sob demanda.

Desta maneira, a informação definida nos perfis de execução é também consultada quando da carga de serviços sob demanda, assim, a estratégia adaptativa para carga dos serviços acontece tanto na inicialização do nodo, quanto após este já estar em operação e precisar instalar um novo serviço.

Esta política para carga dos serviços é disponibilizada por um núcleo mínimo do EXEHDA, o qual é instalado em todo EXEHDAnodo que for integrado ao ISAMpe. Este núcleo é formado por dois componentes (figura 4.5):

- **ProfileManager:** interpreta a informação disponível nos perfis de execução e a disponibiliza aos outros serviços do *middleware*. Cada EXEHDAnodo tem um perfil de execução individualizado;

- **ServiceManager**: realiza a ativação dos serviços no EXEHDAnodo a partir das informações disponibilizadas pelo ProfileManager. Para isto, carrega sob demanda o código dos serviços do *middleware*, a partir do repositório de serviços que pode ser local ou remoto, dependendo da capacidade de armazenamento do EXEHDAnodo e da natureza do serviço.

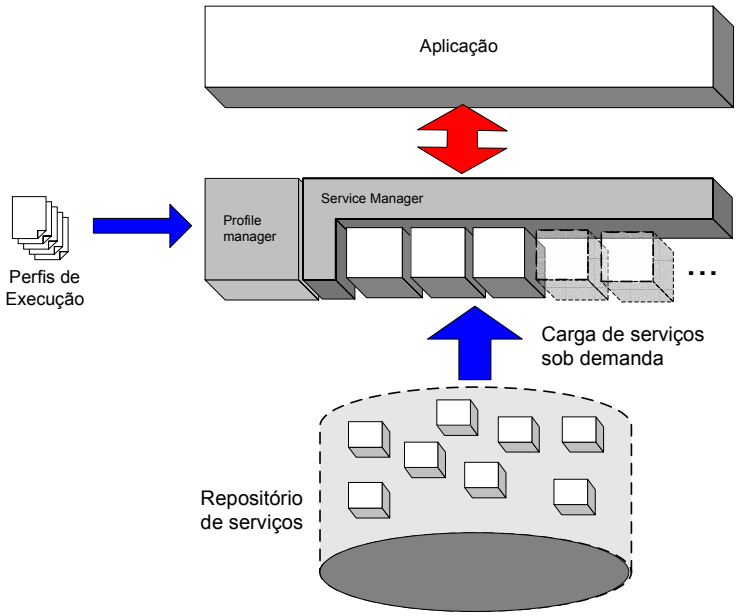


Figura 4.5: Organização do núcleo do EXEHDA

4.4.1 Definindo perfis de execução do EXEHDA

Cada EXEHDAnodo tem o seu perfil de execução do *middleware* especificado através de um documento XML. Este documento associa nomes simbólicos de serviços a componentes que implementam a interface definida para os mesmos. O nome canônico de um serviço é padronizado, sendo determinado pela interface exportada.

É possível, ainda, definir, no perfil de execução para um determinado nodo, propriedades cujos valores sejam recuperados em tempo de processamento, com o intuito de personalizar o funcionamento de um serviço. O formato genérico do documento XML que descreve o perfil de execução do *middleware* é apresentado na figura 4.6. Por sua vez, um exemplo de documento instanciado para definição de perfil de execução do EXEHDA pode ser visto na figura 4.7.

```
<EXEHDA>
 <PROFILE name="profileName">
  <SERVICE name="sName" impl="className" loadPolicy="boot"|"demand">
   <PROP name="paramName" value="paramValue" />
  </SERVICE>
 </PROFILE>
</EXEHDA>
```

Figura 4.6: Formato do documento de definição de perfil de execução do EXEHDA

Na figura 4.7, um bloco <PROFILE> define um perfil de execução, sendo o valor do atributo “*name*” quem define o nome associado àquele perfil de execução. Internamente ao bloco <PROFILE>, blocos <SERVICE> são utilizados para descrição dos serviços que integram aquele perfil de execução específico.

```
<EXEHDA>
  <PROFILE name="desktop-node">

    <SERVICE name="logger" loadPolicy="boot">
      <PROP name="impl" value="org.isam.exehda.services.logging.BasicLogger"/>
      <PROP name="logLevel" value="5000"/>
    </SERVICE>

    <SERVICE name="worb" loadPolicy="boot">
      <PROP name="impl" value="org.isam.exehda.services.worb.WorbImpl"/>
    </SERVICE>

    <SERVICE name="cib" loadPolicy="boot">
      <PROP name="impl" value="org.isam.exehda.services.cib.CibImplClient"/>
      <PROP name="contactAddress" value="//143.54.7.137/cib" />
    </SERVICE>

    <SERVICE name="bda" loadPolicy="boot">
      <PROP name="impl" value="org.isam.exehda.services.bda.BdaImplClient"/>
      <PROP name="bdaHost" value="lunaris.inf.ufrgs.br" />
    </SERVICE>

    <SERVICE name="avu" loadPolicy="demand">
      <PROP name="impl" value="org.isam.exehda.services.avu.AvuImplClient"/>
      <PROP name="avuHost" value="lunaris.inf.ufrgs.br" />
    </SERVICE>

    <SERVICE name="executor" loadPolicy="boot">
      <PROP name="impl" value="org.isam.exehda.services.primos.ExecutorImpl"/>
    </SERVICE>

    <SERVICE name="gatekeeper" loadPolicy="boot">
      <PROP name="impl" value="org.isam.exehda.services.GatekeeperService"/>
      <PROP name="port" value="29901"/>
    </SERVICE>

    <SERVICE name="objectseed" loadPolicy="demand">
      <PROP name="impl" value="org.isam.exehda.services.primos.ObjectSeedImpl"/>
    </SERVICE>

    ...

  </PROFILE>
</EXEHDA>
```

Figura 4.7: Exemplo de documento de definição de perfil de execução do EXEHDA

No tocante ao bloco <SERVICE>, o atributo “*name*” indica o nome canônico do serviço. Ainda, o atributo “*impl*” especifica o componente (uma classe Java na versão atual do protótipo do EXEHDA) que deve ser utilizado como implementação para o serviço, enquanto o atributo “*loadPolicy*” define a política de carga que o núcleo do EXEHDA deve utilizar para o serviço. Nesse sentido, o atributo “*loadPolicy*” pode assumir os valores: (*i*) “*boot*”, indicando que o serviço deve ser carregado durante o *bootstrap* do *middleware,* e (*ii*) “*demand*”, significando que o serviço deve ser carregado quando for utilizado pela primeira vez.

Os elementos <PROP> são utilizados dentro de um bloco <SERVICE> para definir propriedades que poderão ser recuperadas em tempo de execução pelo serviço, para personalização de sua execução.

A parametrização dos serviços, via perfil de execução, também tem o papel de suportar a visão unificada da EXEHDAbase, repassando para os serviços em execução nos EXEHDAnodos à estratégia específica que deve ser utilizada para localização da

instância celular daquele serviço. Duas estratégias estão previstas: (*i*) uma referência para o dado armazenado na CIB, ou (*ii*) uma referência direta para o nodo que hospeda a instância celular daquele serviço. Esta segunda estratégia, via de regra, é reservada para indicação da instância celular da CIB (*Cell Information Base*, vide seção 4.5.2).

4.4.2 Nomenclatura empregada pelo EXEHDA

Um aspecto significativo da concepção do EXEHDA é empregar identificadores únicos no escopo do ISAMpe. Considerando a demanda por escalabilidade inerente à Computação Pervasiva, tais identificadores únicos são construídos por composição, tendo por base o identificador do EXEHDAnodo. Os principais identificadores empregados no EXEHDA são:

- **HostId**: identifica unicamente um EXEHDAnodo no ISAMpe. É composto de um nome de célula e um identificador numérico de 32 bits único no escopo daquela célula. Por exemplo: HostId:{Id=0x1A07B159, cell= “inf.ufrgs.br”}. Assume-se que cada EXEHDAcel tem um nome distinto, o qual pode ser derivado de seu domínio na Internet;
- **ApplicationId**: identifica unicamente uma aplicação em execução no ISAMpe. É composto por um identificador único de 512 bits, agregado ao HostId do EXEHDAnodo a partir do qual a aplicação foi disparada. Por exemplo: ApplicationId:{Id=<número 64 bytes>, HostId:{Id=0x1A07B159, cell=”inf.ufrgs.br”}};
- **ObjectId**: identifica unicamente um OX (Objeto eXehda, seção 4.5.3) no escopo do ISAMpe. É composto por um identificador inteiro de 16 bits acrescido de um hash também de 16 bits e do HostId do nodo onde o OX foi inicialmente instanciado. Observe-se que o identificador da aplicação ao qual o OX pertence não é incluído explicitamente no ObjectId, contudo esta informação é considerada no cálculo do hash e disponibilizada na forma de um atributo do OX, acessível por intermédio do serviço *OXmanager*.

Outro aspecto importante no EXEHDA é a diferenciação entre os conceitos de política e heurística. É entendida como política um conjunto incompleto de parâmetros que influenciará a operação de um determinado componente em tempo de execução. Tipicamente, uma política resume o conhecimento do programador sobre um aspecto específico do comportamento da aplicação.

Por sua vez, é entendida como heurística o algoritmo implementado por um componente e utilizado para tomada de decisão, a qual considera, freqüentemente, um modelo simplificado do ambiente. Em tempo de execução uma heurística pode se valer dos parâmetros definidos em uma política, assim como de outras informações extraídas de forma dinâmica do ambiente de execução para tomada de decisão.

Na perspectiva do EXEHDA estes dois conceitos estão instanciados nas seguintes possibilidades:

- **políticas de adaptação da aplicação**: definem em alto nível parâmetros que influenciarão o comportamento dos serviços do subsistema de Adaptação e Reconhecimento de Contexto;

- **heurísticas de aplicação**: tipicamente estas heurísticas estão associadas ao modelo de computação do paradigma de programação utilizado no desenvolvimento da aplicação. Assim, todas as aplicações desenvolvidas em uma determinada linguagem podem compartilhar uma mesma heurística de aplicação. Estas heurísticas são responsáveis por traduzir e complementar as políticas de aplicação, as quais são definidas em alto nível, para requisições mais elementares, a serem entregues aos respectivos serviços de adaptação;
- **políticas do middleware**: descrevem parâmetros que influenciam todas as aplicações em execução;
- **heurísticas do middleware**: combinam para a tomada de decisão, as informações geradas pela heurística da aplicação, com as informações advindas das políticas do *middleware*.

A combinação entre políticas e heurísticas, de aplicação e do middleware, conforme apresentado na figura 4.8, materializam o modelo de adaptação colaborativa multinível introduzido na seção 3.5.3. Esta organização baseada em duas políticas e duas heurísticas, reflete-se tanto na gerência das adaptações funcionais e não-funcionais (vide seção 4.6).

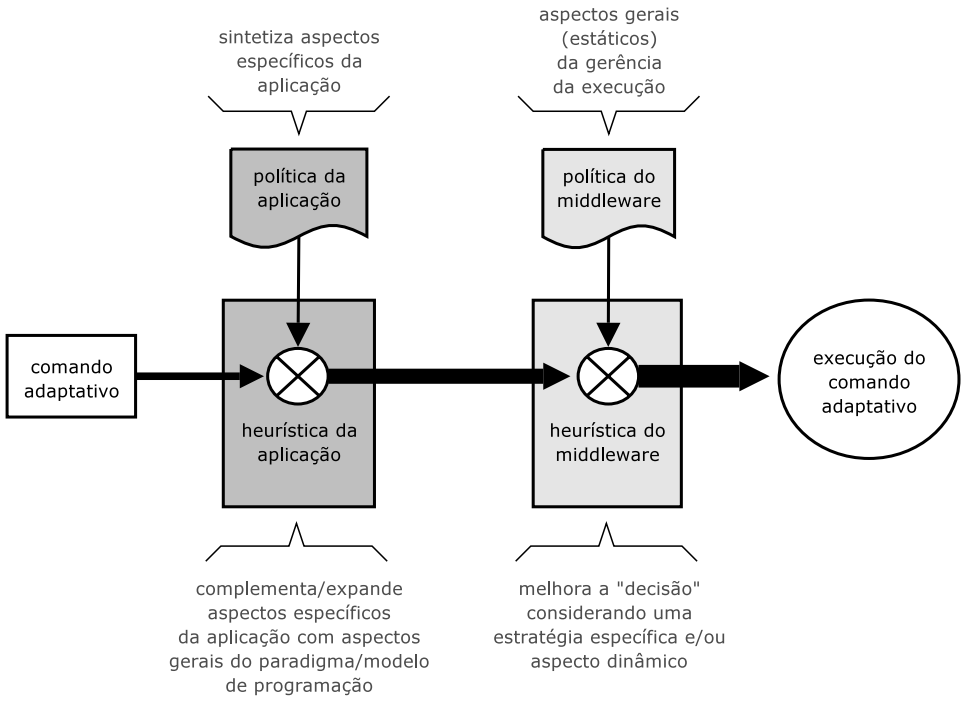


Figura 4.8: O modelo colaborativo multinível na execução de comandos adaptativos

4.5 Subsistema de execução distribuída

O Subsistema de Execução Distribuída é responsável pelo suporte ao processamento distribuído no EXEHDA. No intuito de promover uma execução efetivamente pervasiva, este subsistema interage com outros subsistemas do EXEHDA. Em

específico, interage com o subsistema de reconhecimento de contexto e adaptação, de forma a prover comportamento distribuído e adaptativo às aplicações ISAM.

4.5.1 Executor

O serviço *Executor* acumula as funções de disparo de aplicações, e de criação e migração de seus objetos. Na implementação destas funções é empregada a instalação de código sob demanda. Para tal, interage com os serviços CIB (seção 4.5.2) e BDA (seção 4.8.1). A interface do serviço *Executor* e classes associadas podem ser vistas na figura 4.9.

A interface do serviço *Executor* define métodos que controlam os ciclos de vida das aplicações e dos objetos. Os métodos `startApplication` e `exitApplication` são utilizados no disparo e encerramento de aplicações. O método `currentApplication`, por sua vez, permite recuperar a identificação única da aplicação (ApplicationId) no EXEHDA.

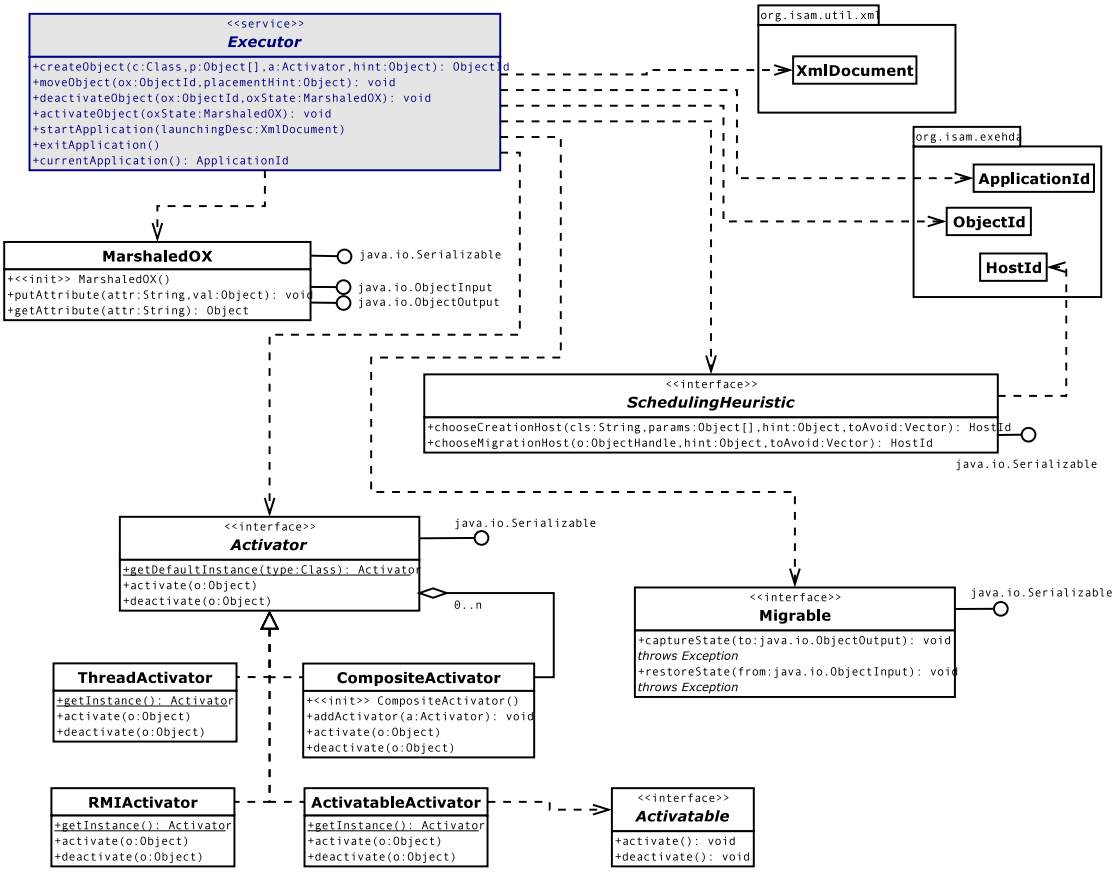


Figura 4.9: Diagrama de classes do serviço *Executor*

O método `createObject` é responsável pela instanciação remota OXs (Objetos eXehda). Estes objetos são explicitamente gerenciados pelo *middleware* (seção 4.5.3). O método `moveObject` permite modificar o posicionamento físico de um OX previamente instanciado via `createObject`. O implementador de um objeto caracteriza o potencial de mobilidade de um objeto pela implementação da interface `Migrable`, a qual define métodos para captura e restauração do estado da execução do objeto. A operação de captura e/ou restauração do estado do objeto é um requisito básico para viabilizar a sua migração.

Por fim, os métodos `deactivateObject` e `activateObject` permitem respectivamente suspender e restaurar um objeto em execução. Neste sentido, o objeto suspenso e seus atributos de execução são temporariamente armazenados em uma estrutura denominada `MarshaledOX`. Esta funcionalidade é utilizada na implementação da migração de código, onde a estrutura `MarshaledOX` é passada como parâmetro dos métodos definidos na interface `Migrable`, como também pelo serviço *SessionManager* quando do salvamento e/ou recuperação da sessão do usuário (seção 6.2.1).

O serviço *Executor* emprega um modelo flexível, baseado no padrão de projeto Strategy [GAM 97], tanto para a ativação dos objetos instanciados, quanto para a seleção do nodo que receberá o objeto. O padrão de projeto Strategy materializa uma determinada classe de problema na forma de uma interface, permitindo que seja alterada a abordagem específica adotada na solução do mesmo, isto é, na implementação da interface definida, sem exigir a modificação nos outros componentes do sistema.

No que se refere à ativação de objetos no EXEHDA, o padrão de projeto Strategy apresenta-se na forma de objetos ativadores (*Activators*), providos como parâmetros da instanciação remota. Os objetos ativadores são responsáveis pelo disparo de *threads* associadas ao objeto instanciado, assim como pela inicialização de outros recursos externos ao objeto instanciado, e necessários à operação do mesmo como, por exemplo, bibliotecas que precisem ser carregadas.

Por sua vez, na perspectiva da seleção do nodo destino para as operações de instanciação remota ou migração, o padrão de projeto Strategy encontra-se refletido na heurística de escalonamento (*SchedulingHeuristic*), para a qual é delegada a seleção do nodo destino da operação de instanciação remota ou migração.

Tipicamente, a heurística de escalonamento, considera a política de adaptação não funcional (`scheduling.xml`), e realiza uma transformação da requisição de instanciação remota ou migração em uma especificação abstrata de recurso (vide *Scheduler*, seção 4.6.5). Um exemplo de política de adaptação não funcional é apresentada na seção 5.2, que aborda um caso prático de integração do EXEHDA com uma linguagem desenvolvida para o cenário da computação pervasiva.

A construção da especificação abstrata de recurso consiste na incorporação de parâmetros definidos na política de escalonamento da aplicação (especificada em tempo de desenvolvimento) quando de uma instanciação remota ou migração de objetos.

Estes parâmetros são repassados ao serviço *Scheduler*, integrante do subsistema de adaptação, para a efetiva seleção do nodo destino. Desta forma, a execução distribuída adquire uma primeira dimensão de adaptação.

Uma vez selecionado o nodo destino, a instância local do serviço *Executor* acessa a instância remota deste mesmo serviço, em execução no nodo destino, para conclusão da operação. Este acesso remoto é intermediado pelo subsistema de comunicação do EXEHDA.

4.5.2 Cell Information Base - CIB

O serviço *Cell Information Base* (CIB) implementa a base de informações da célula. Sua principal funcionalidade está relacionada à manutenção da infra-estrutura distribuída que forma o ISAMpe. A interface do serviço CIB e classes associadas podem ser vistas na figura 4.10.

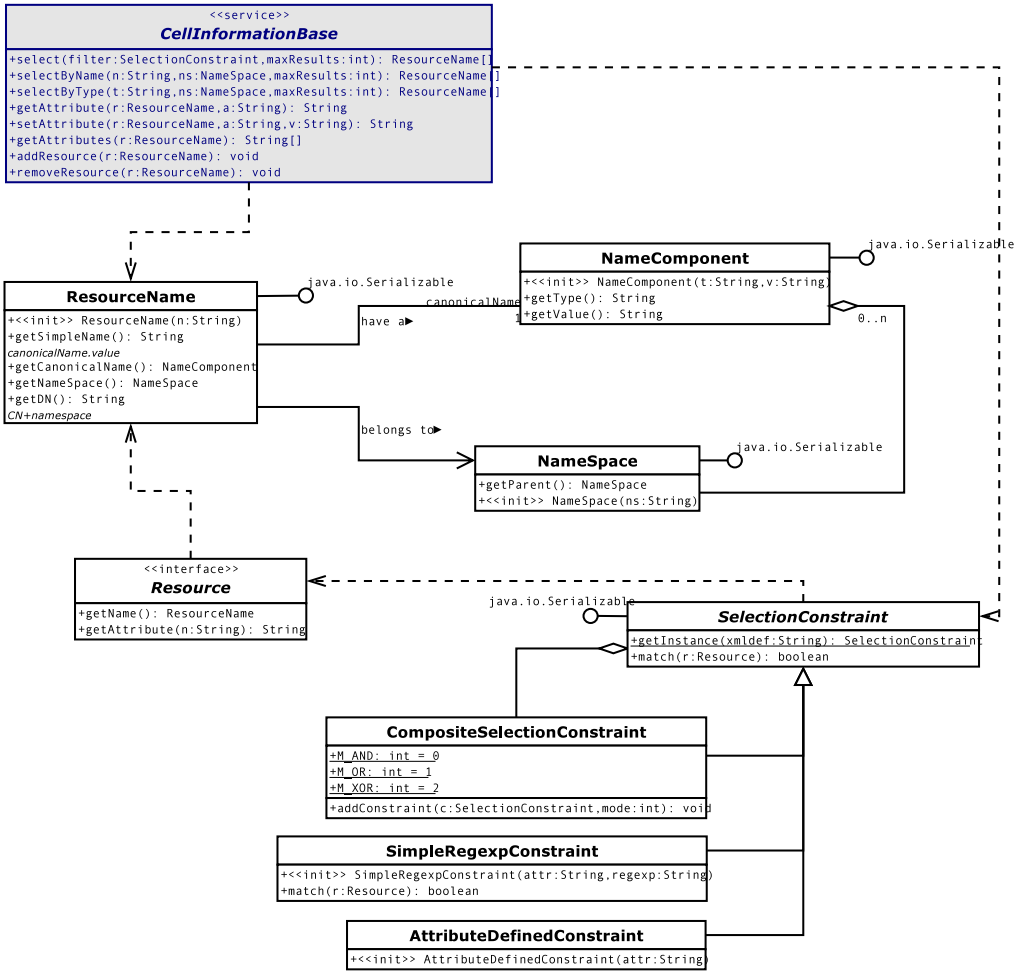


Figura 4.10: Diagrama de classes do serviço CIB

Este serviço mantém os dados estruturais da EXEHDAcel, tais como, informações sobre os recursos que a compõem, informação de vizinhança, e atributos que descrevem as aplicações em execução. Particularmente, quanto às aplicações em execução, os atributos registrados pelo serviço *Executor* quando do disparo da aplicação incluem:

- identificação do proprietário (*owner*);
- referência para o código da aplicação no serviço BDA;
- descritor de disparo da aplicação.

O serviço CIB é ainda responsável pela manutenção das informações referentes aos usuários registrados na célula. Estas informações incluem um certificado assinado que mantém a chave pública do usuário. Este certificado é, posteriormente, utilizado para autenticação do usuário, como por exemplo no procedimento de login (vide seção 6.2.1). A opção atual é pelo uso do formato de certificados X509 [HOU 2002], o qual consiste de uma tecnologia de domínio público especificada pela IETF [IET 2002].

A organização das informações na CIB é baseada em espaços de nomes tipados. Nesta proposta, cada componente do nome é constituído por um tipo e um valor. São minimizados, desta forma, potenciais conflitos usuais em espaços de nomes cujos componentes são baseados somente em um atributo valor, por exemplo o [DNS 2002] empregado na Internet. A abordagem proposta para o EXEHDA permite integrar diferentes tipos de recursos sob uma mesma estrutura organizacional.

Embora não seja relevante para a especificação, é significativo ressaltar que o armazenamento dos dados na base de informações da célula pode ser feito utilizando-se diversas alternativas de armazenamento (*backends)*; desde soluções baseadas em tabelas hash em memória, até sofisticados serviços de diretório como o LDAP ou X500 [OLD 2003].

Considerando a disseminação de serviços de diretórios baseados em LDAP, o uso deste mecanismo na implementação da CIB potencializa uma integração simbiótica da CIB com mecanismos tradicionais de manutenção e gerência de recursos em rede.

A interface do serviço *CellInformationBase*, ilustrada na figura 4.10, disponibiliza métodos com duas naturezas de atuação: (*i*) registro e remoção de recursos e/ou seus atributos associados, (*ii*) pesquisa na sua base de informações por recursos a partir de um conjunto de atributos de interesse.

4.5.3 OXManager

A abstração OX - Objeto eXehda -, provida pelo *middleware* às aplicações, consiste em uma instância de objeto, criada por intermédio do serviço *Executor*, à qual pode ser associada meta-informação em tempo de execução. No caso da abstração OX, esta meta-informação ocorre na forma de atributos, i.e., pares <nome, valor>, sendo que "nome" é uma cadeia de caracteres ASCII que descreve o atributo, podendo "valor" ser um conteúdo genérico, inclusive de natureza binária.

A gerência e manutenção da meta-informação associada a um OX é atribuição do serviço *OXManager*. Este serviço confere às operações de consulta e atualização dos atributos do OX o necessário caráter pervasivo, permitindo que estes sejam acessados a partir de qualquer nodo do ISAMpe. A interface do serviço *OXManager* é apresentada na figura 4.11.

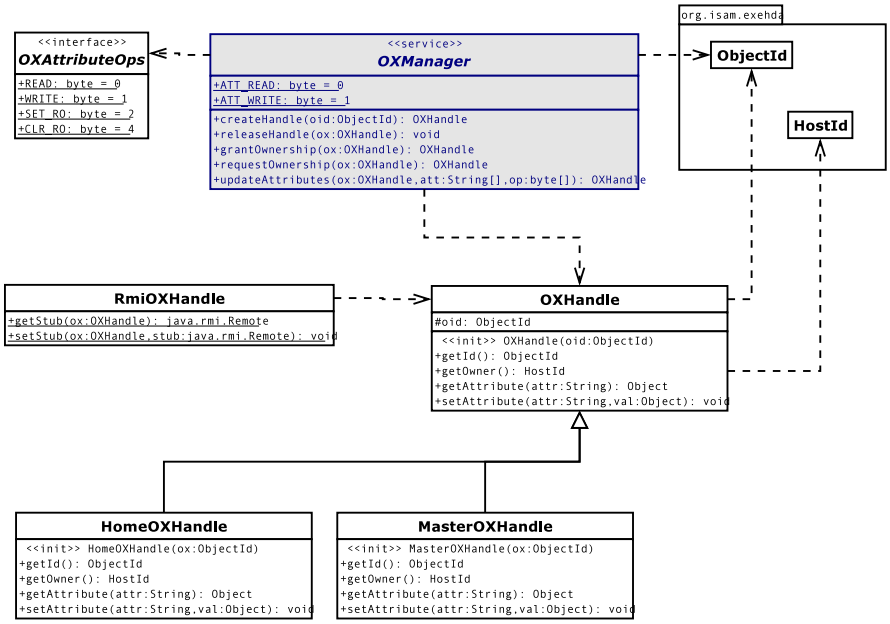


Figura 4.11: Diagrama de classes do serviço *OXManager*

O suporte à meta-informação do OX, na perspectiva escalável requerida pela Computação Pervasiva, é baseado em estruturas denominadas OXHandles e operações

para manipulação destes OXHandles, definidas na interface do serviço *OXManager*, conforme caracterizado a seguir.

Um OXHandle pode assumir um de três papéis: master, home e proxy. O proxy OXHandle representa o caso base (*default*), consistindo de um procurador para o efetivo acesso ao master e home OXHandles. O OXHandle inclui, ao menos, o ObjectId associado ao OX.

Por sua vez, o master OXHandle corresponde à cópia principal dos atributos do OX. O mesmo é criado durante processo de ativação, conduzido pelo serviço *Executor* quando da instanciação do OX. Para isso, é empregada a chamada `createHandle` ao serviço *OXManager* do EXEHDAnodo que está recebendo a instanciação.

Tipicamente, o master OXHandle é mantido no mesmo nodo que o OX ao qual está associado, ou seja, este acompanha o OX nas migrações de forma a otimizar acessos em escrita. Contudo, esta não é uma regra rígida, podendo o master OXHandle ser deslocado para outro EXEHDAnodo com vistas à economia de recursos em um EXEHDAnodo, por exemplo com uma capacidade de memória restrita. Desta forma, faz-se necessário, um mecanismo que permita localizar o master OXHandle associado a um determinado OX antes que possa ser possível acessar seus atributos. Tal mecanismo é provido via o home OXHandle.

O home OXHandle é sempre armazenado na instância celular do serviço *OXManager* correspondente à EXEHDAcel na qual o OX foi inicialmente instanciado (célula home). A informação de qual é a célula home é um pseudo-atributo típico de um OX, o qual é derivado implicitamente a partir do ObjectId daquele OX. O home OXHandle implementa *caching* dos atributos não mutáveis do OX como ‘oid’, que mantém o ObjectId do OX, e ApplicationId, que identifica a aplicação à qual o OX pertence. Adicionalmente, mantém no attributo ‘owner’ a identificação da instância de *OXManager* que atualmente gerencia o master OXHandle daquele OX.

Do ponto de vista das aplicações que empregam os OXHandles para acesso à meta-informação associada a um dado OX, esta diferenciação em master, home e proxy não é significativa (perceptível). Tipicamente, uma aplicação emprega os métodos `createHandle` e `releaseHandle` para alocar e liberar um OXHandle que lhes permita o acesso aos atributos do OX. Implicitamente, o serviço *OXManager* do EXEHDAnodo local interage com a instância celular e esta com as demais células, atuando sobre o posicionamento do master OXHandle e atualizando o home OXHandle conforme necessário. Para isto, utiliza os métodos `grantOwnership` e `requestOwnership`. Estas operações permitem influenciar a decisão de posicionamento físico do master OXHandle, de forma que o *middleware* possa otimizar os acessos em leitura e escrita aos atributos, ao mesmo tempo em que garante a consistência das atualizações.

A abstração OX provida pelo EXEHDA é útil no estabelecimento de mapeamentos entre o modelo de computação, utilizado no nível da linguagem de programação, e a funcionalidade oferecida pelo *middleware*. Um caso que ilustra tal situação é o serviço *BeingManager*, que integra o suporte à linguagem ISAMadapt e utiliza o mecanismo de atributos de execução provido pelo serviço *OXManager* para manutenção da hierarquia lógica (que é dinâmica) dos entes que compõem uma aplicação ISAMadapt (vide capítulo 5).

4.5.4 Discoverer

O serviço de descoberta de recursos – *Discoverer* - do EXEHDA é responsável pela localização de recursos especializados no ISAMpe a partir de especificações abstratas dos mesmos. As especificações caracterizam o recurso a ser descoberto por meio de atributos e seus respectivos valores. Adicionalmente, a requisição de descoberta de recurso incorpora um parâmetro que define a amplitude da pesquisa. Nesse sentido, os seguintes valores estão definidos:

0: próprio nodo;
1: segmento local;
2: célula local;
3: vizinhança estática da célula local;
4: vizinhança completa da célula local, nível 1;
5: vizinhança completa da célula local, nível 2.

A linguagem para descrever uma requisição de descoberta de recursos está apresentada na figura 4.12. A utilização de um documento XML para especificar a requisição de descoberta de recursos, além de potencializar a interoperabilidade com outros sistemas, mantém o formato adotado na definição dos documentos empregados em outras instâncias do EXEHDA. Registre-se que esta linguagem constitui uma solução inicial, e seu melhoramento constitui tema de pesquisa em andamento (vide trabalho [SCH 2004]). Contudo, a premissa perseguida é a de que uma modificação da linguagem atualmente empregada na pesquisa não altere a interface projetada para o serviço *Discoverer*.

```
<QUERY scope=”k”>
   <ATT n=”<nome_atributo_1>” v=”valor_atributo_1”/>
   <ATT n=”<nome_atributo_2>” v=”valor_atributo_1”/>
   ...
   <ATT n=”<nome_atributo_n>” v=”valor_atributo_n”/>
</QUERY>
```

Figura 4.12: Linguagem de pesquisa empregada no serviço *Discoverer*

A interface do serviço *Discoverer* (figura 4.13) define operações para o registro/remoção de recursos e para a pesquisa por recursos que atendam a um determinado critério (disponibilizem um conjunto de atributos de interesse), métodos `register`, `remove` e `find` respectivamente. Os recursos registrados no *Discoverer* são catalogados no serviço CIB (*Cell Information Base*), ficando a partir de então visíveis no ISAMpe.

O registro de recurso definido adota o conceito de *leases*, i.e., deve ser periodicamente renovado, o que ocorre através do método `renew`; do contrário, o recurso é automaticamente removido do conjunto de recursos registrados como disponíveis no serviço CIB.

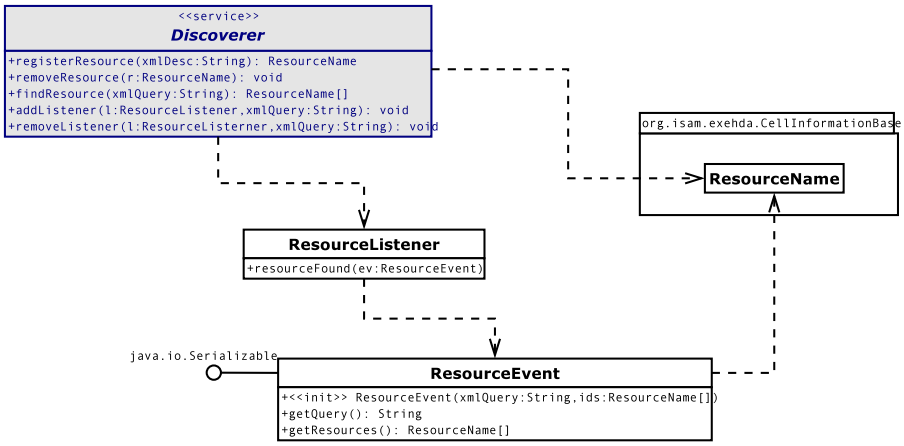


Figura 4.13: Diagrama de classes do serviço *Discoverer*

Na resolução de uma requisição para descoberta de recursos, o serviço *Discoverer* interage com o serviço CIB da sua EXEHDAcel, tentando satisfazer a mesma no escopo local da célula.

No EXEHDA a localização de um recurso não implica qualquer tipo de alocação ou reserva do mesmo. O serviço responsável pelo controle e implementação das políticas de alocação, i.e., a gerência de quais aplicações e/ou usuários tem permissão de alocar que recursos e durante que períodos, é o *ResourceBroker*, abordado na próxima seção.

Quando o escopo da pesquisa receber valor 4 ou superior, e uma requisição não puder ser resolvida em âmbito celular, a busca pelo recurso irá promover a interação do *Discoverer* local com o serviço *ResourceBroker* das células vizinhas.

4.5.5 ResourceBroker

O controle da alocação de recursos às aplicações no EXEHDA é desempenhado pelo serviço *ResourceBroker*, o qual atende tanto requisições originárias da própria EXEHDAcel quanto oriundas de outras células do ISAMpe.

A interface do serviço *ResourceBroker*, apresentada na figura 4.14, define operações de duas naturezas: (*i*) localização (`findResource`) e (*ii*) alocação de recursos (`allocResource`). A atuação do serviço *ResourceBroker* determina o subconjunto dos recursos locais a ser visível ao universo externo a célula.

No tratamento das requisições de localização e alocação, o serviço *ResourceBroker* interage com os serviços *Discoverer* e *Scheduler* internos à sua célula para, respectivamente, localização de recurso especializado e alocação de nodo de processamento. A figura 4.15 exemplifica a operação inter-celular do serviço *ResourceBroker* e o relacionamento deste com o serviço *Scheduler* para alocação de um EXEHDAnodo, o qual hospedará um OX a ser instanciado.

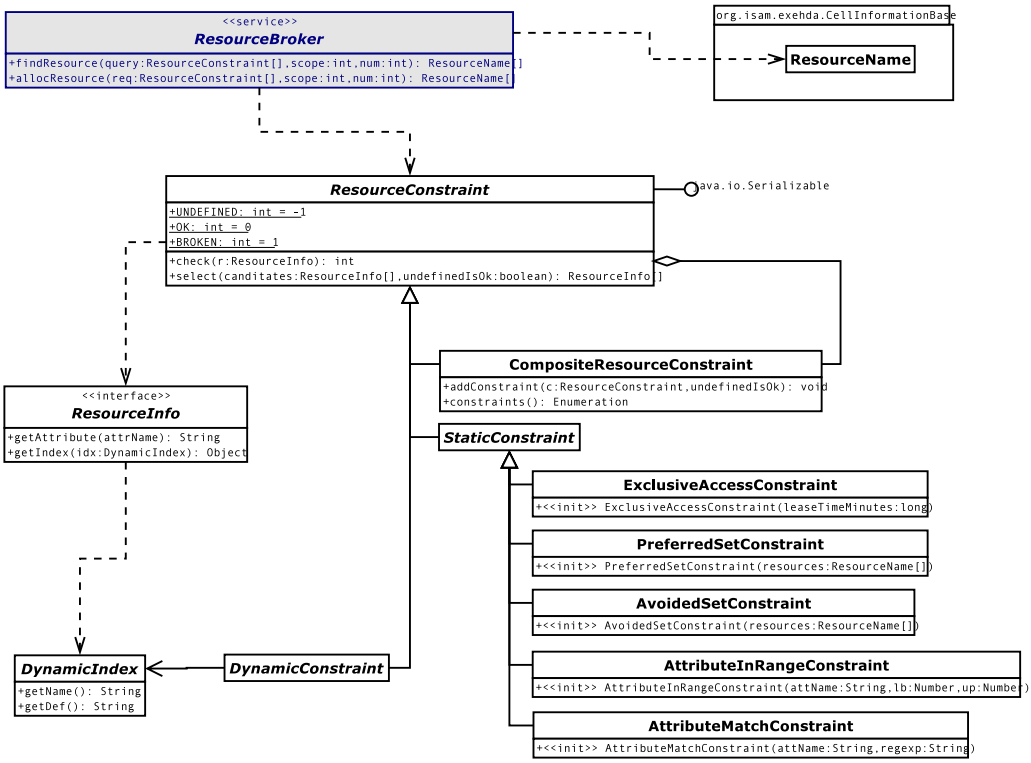


Figura 4.14: Diagrama de classes do serviço *ResourceBroker*

A especificação das características de um recurso a ser localizado ou alocado ocorre na forma de restrições (ResourceConstraint) que podem ser estáticas ou dinâmicas. Restrições estáticas incluem: (*i*) a especificação de atributos que devem ser providos pelo recurso, ou (*ii*) um conjunto de recursos preferenciais, ou ainda (*iii*) um conjunto de recursos que deve ser evitado. Restrições dinâmicas incluem a definição de um índice dinâmico para o recurso, tipicamente construído a partir da informação extraída pela monitoração (seção 4.6.1). Efetivamente, o *ResourceBroker* não trata índices dinâmicos mas delega esta operação ao serviço apropriado (*Discoverer* ou *Scheduler*).

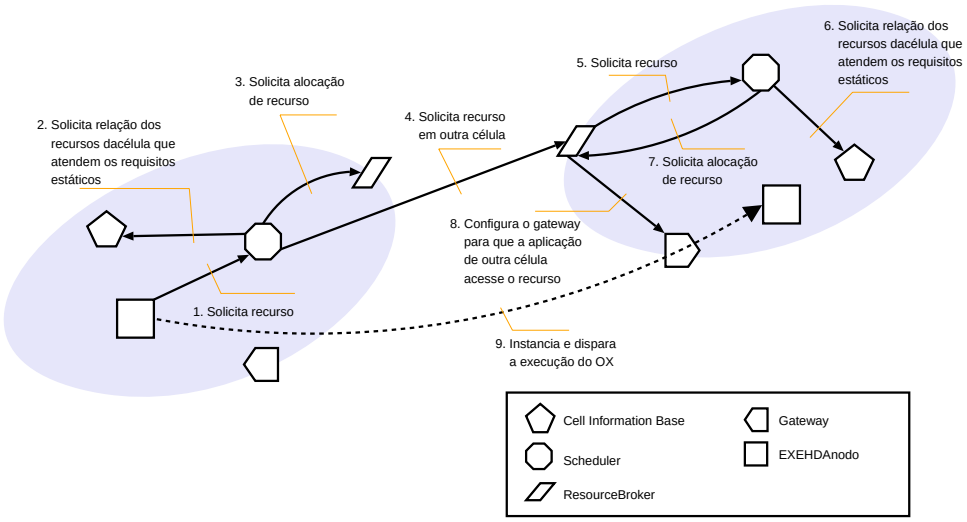


Figura 4.15: Alocação inter-celular de recursos de processamento

No que se refere à alocação de nodos de processamento, esta pode se dar em dois modos: (*i*) compartilhado ou (*ii*) exclusivo para aplicação. O suporte a um ou ambos destes modos é uma característica da implementação específica do serviço *ResourceBroker* em uso ou da política de alocação vigente na célula. Observe-se que o acesso exclusivo a um recurso, por uma aplicação, tem como parâmetro o tempo máximo de duração da alocação. Como resultado de uma alocação com sucesso, o serviço *ResourceBroker*, parametriza o serviço Gateway local, alterando a "permeabilidade" da célula, de forma que a aplicação que requisitou o recurso alocado possa ter acesso ao mesmo. Ao término da aplicação, como parte dos procedimentos de finalização da mesma, este acesso é revogado.

Por outro lado, no que se refere à localização de recursos especializados, o *ResourceBroker* preserva a semântica anteriormente definida para o serviço *Discoverer*, pela qual requisições de localização de um recurso não implicam a alocação ou reserva do recurso.

4.5.6 Gateway

O serviço *Gateway* faz a intermediação das comunicações entre os nodos externos à célula e os recursos internos a ela. Da sua ação integrada com o *ResourceBroker* decorre o controle de acesso aos recursos de uma EXEHDAcel. Pode-se sintetizar que:

controle de acesso entre células = *Gateway* + *ResourceBroker*.

A interface do serviço *Gateway*, ilustrada na figura 4.16, define três métodos direcionados à concessão, revogação e renovação de permissões de acesso. Nesse sentido, o método `grant` libera o acesso a um conjunto de EXEHDAnodos para uma dada aplicação, pelo período de tempo determinado quando da sua chamada. Caso seja necessário estender a validade de uma dada permissão, esta operação é efetuada pelo uso do método `renew`. Por outro lado, caso seja preciso antecipar o término da validade de uma permissão, e.g., caso a aplicação em questão tenha terminado seu processamento, isto é possível através do método `drop`.

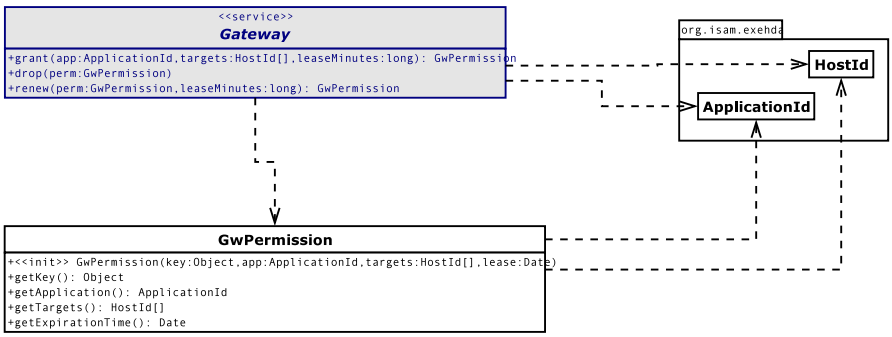


Figura 4.16: Diagrama de classes do serviço *Gateway*

Inicialmente nenhum recurso da célula é feito visível ao universo externo à mesma. À medida que recursos vão sendo alocados junto ao *ResourceBroker*, a condição de "permeabilidade" da célula é alterada, refletindo-se na operação do serviço *Gateway*. Esta permeabilidade, gerenciada pelo *Gateway*, é que faculta, efetivamente, uma aplicação distribuída entre várias células empregar os recursos do ISAMpe.

O serviço *Gateway* mantém uma tabela de pares <*aplicação, recurso*>, a qual registra que recursos são visíveis para quais aplicações. Esta tabela é dinamicamente alterada a partir da interação com o *ResourceBroker*.

4.5.7 StdStreams

O serviço *StdStreams* do *middleware* provê o suporte ao redirecionamento dos *streams* padrões de entrada, saída e erro. Sua funcionalidade se dá numa perspectiva por aplicação, sem a necessidade de modificação no código da mesma. Para isto, define que cada aplicação em execução em um determinado EXEHDAnodo possui um componente *Console* individualizado, o qual agrupa os três *streams* padrões. A interface do serviço *StdStreams* é apresentada na figura 4.17.

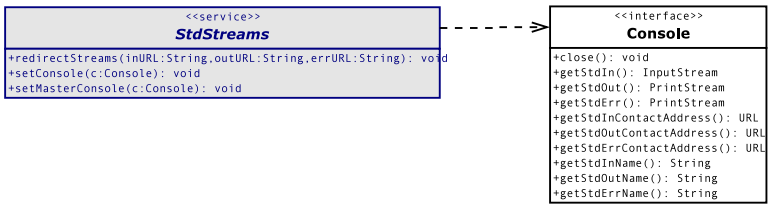


Figura 4.17: Diagrama de classes do serviço *StdStreams*

O comportamento padrão é associar à aplicação um objeto console nulo, significando que operações de leitura retornarão EOF (fim de arquivo) e operações de escrita serão descartadas sem notificação. A aplicação pode modificar este comportamento, personalizando o *Console* utilizado em cada nodo explicitamente, ou através de atributos de execução definidos na forma de URLs providas por ocasião do disparo da aplicação (vide seção 6.2.2). Nesse último caso, a parametrização do *StdStreams* é feita pelo serviço *Executor* quando este inicia a aplicação.

Observe-se que as operações de entrada e saída via *StdStreams* são relativas ao EXEHDAnodo local. Contudo, é possível a aplicação especificar um componente *Master Console*. Este componente é entendido como um centralizador das operações de entrada e saída via *StdStreams*, sendo aplicável a todos os EXEHDAnodos onde a aplicação não tenha explicitamente personalizado o *Console* local via `setConsole`. Nesse modo de operação, os consoles locais em cada EXEHDAnodo são substituídos por *proxies* que redirecionam as operações de entrada e saída para as URL's definidas no componente *Master Console*.

4.5.8 Logger

A funcionalidade de registro de rastro de execução (*logging*) é amplamente empregada em sistemas computacionais [STI 99]. Na fase de desenvolvimento, pode ser empregada para depuração de um programa auxiliando a identificar comportamentos errôneos ou imprevistos (gargalos de execução - *bottlenecks*). Por sua vez, considerando a segurança dos sistemas computacionais, esta funcionalidade é freqüentemente empregada para registro de operações importantes e/ou críticas realizadas, facilitando a identificação de situações de intrusão, ou de uso indevido do sistema.

Para atendimento destes aspectos, no EXEHDA é disponibilizado o serviço *Logger*, cuja interface é apresentada na figura 4.18. O *Logger*, durante os períodos de desenvolvimento, tem constituído uma facilidade importante para depuração dos demais serviços, assim como para geração de dados sobre o perfil da utilização dos mesmos.

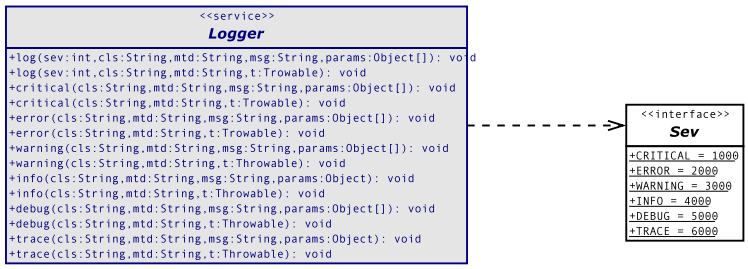


Figura 4.18: Diagrama de classes do serviço *Logger*

O serviço *Logger* disponibiliza métodos para registro de mensagens, considerando que tais mensagens são classificadas segundo níveis de prioridade. Além do método básico `log`, através do qual o usuário do serviço pode explicitar o nível de prioridade das mensagens, inclui também facilidades para o registro de mensagens com níveis de prioridade mais usualmente encontrados: situação crítica, erro, alerta, mensagem informativa e depuração.

4.5.9 Dynamic Configurator - DC

O objetivo do serviço *DynamicConfigurator* (DC) do EXEHDA é realizar a configuração do perfil de execução do *middleware* em um determinado EXEHDAnodo de forma automatizada. A interface do serviço *DynamicConfigurator* é apresentada na figura 4.19.

A configuração automática dos EXEHDAnodos é consolidada com a participação do componente *ServiceManager*, o qual controla o processo de *bootstrap* do *middleware*. Assim, para que um EXEHDAnodo seja configurado de forma automática, o *middleware* é inicializado empregando um perfil de ativação base que contempla referência ao serviço *DynamicConfigurator*. O componente *ServiceManager* identifica esta referência e executa uma chamada ao método `generateProfile` do serviço *DynamicConfigurator*.

Figura 4.19: Diagrama de classes do serviço *DynamicConfigurator*

Inicialmente, o serviço *DynamicConfigurator* executa um processo de detecção básico para identificar as características do EXEHDA. Considerando a implementação do EXEHDA sobre a plataforma Java, esta detecção recupera informações a partir da chamada `System.getProperty` para as seguintes propriedades: `java.version`, `java.vendor`, `os.name`, `os.version`, `os.arch`. Além disso, consulta as seguintes propriedades no perfil base do EXEHDA: `host.name`, `host.id`, `host.net.addr.ip`, `host.net.addr.mac`. Caso tais propriedades não tenham sido definidas no perfil base, executa uma auto-detecção das três primeiras utilizando o suporte presente na classe `java.net.InetAddress` de Java. Adicionalmente, consulta ainda a capacidade de memória do nodo através das chamadas `System.getTotalMemory()` e `System.getFreeMemory()`. De posse de tais informações, o serviço *DynamicConfigurator* monta uma tabela de atributos que descrevem aquele EXEHDAnodo.

Observe-se que o procedimento descrito acima foi concebido de forma a ser portável para todas as implementações do *middleware* sobre a plataforma Java. Esta situação o deixa restrito aos mecanismos comuns às diferentes arquiteturas que coexistem no ISAMpe, apresentando, portanto, algumas limitações . Para tratar esta limitação, quando o perfil base inclui uma instância do serviço *Collector* (vide seção 4.6.1), o DC complementa o conjunto de atributos construído utilizando o procedimento básico de detecção, com informações provenientes dos sensores deste serviço. Desta forma é possível produzir uma descrição mais exata daquele EXEHDAnodo.

Tipicamente, essa informação detectada pela instância do serviço DC local ao EXEHDAnodo é submetida à instância celular do mesmo serviço. Esta utiliza um mecanismo para geração dinâmica do perfil adequado àquele EXEHDAnodo. Nesse sentido, pode empregar regras de configuração combinadas a um repositório de templates de perfis, categorizados por arquitetura. A figura 4.20 exemplifica um conjunto de regras a ser utilizado na seleção do template para um EXEHDAnodo. No exemplo, as regras estão direcionadas para considerar a localização do EXEHDAnodo no ISAMpe. O perfil gerado é então retornado ao componente *ServiceManager*, o qual dispara um novo *bootstrap* do *middleware*, agora considerando a configuração gerada automaticamente.

```
<RULES>
  <RULE attribute="ip" match="143.54.45.*" use="config1.skel" />
  <RULE attribute="hostName" match="*ufrgs.br" use="config2.skel"/>
</RULES>
```

Figura 4.20: Possível configuração da instância celular do serviço DC

O acesso à instância celular do serviço DC pressupõe que a localização da mesma possa ser determinada pela instância nodal (em execução no EXEHDAnodo). Entretanto, na inicialização do *middleware*, a instância nodal do serviço DC não dispõe, a priori, de qualquer informação sobre a célula, nem mesmo a localização da EXEHDAbase.

Com o objetivo de determinar a localização da instância celular do serviço DC é empregado um protocolo baseado em *multicast* sobre IP. Caso, após um determinado número de tentativas, a instância celular do DC não tenha sido localizada, esta informação é requisitada ao Administrador da Célula (vide seção 6.1.2). Neste caso, o fornecimento da localização da EXEHDAbase é feito de forma manual. Todavia, este procedimento é um último recurso e realizado de forma única, pois, depois de configurada a instância nodal do serviço DC em um EXEHDAnodo, este passa a responder às requisições *multicast* dos outros nodos, eliminando a necessidade de que a localização da EXEHDAbase precise ser informada manualmente.

4.6 Subsistema de reconhecimento de contexto e adaptação

O suporte à adaptação no EXEHDA está associado à operação do subsistema de reconhecimento de contexto e adaptação. Este subsistema inclui serviços que tratam desde a extração da “informação bruta" sobre as características dinâmicas e estáticas dos recursos que compõem o ISAMpe, passando pela identificação em alto nível dos elementos de contexto, até o disparo das ações de adaptação em reação a modificações

no estado de tais elementos de contexto. Integram este subsistema os serviços *Collector*, *Deflector*, *ContextManager*, *AdaptEngine* e *Scheduler*.

Particularmente, os serviços *AdaptEngine* e *Scheduler* são responsáveis, respectivamente, pelo controle das adaptações de cunho funcional e não-funcional. Entende-se por adaptação funcional aquela que implica a modificação do código sendo executado. Por sua vez, adaptação não-funcional, é aquela que atua sobre a gerência da execução distribuída, por exemplo o reposicionamento de OXs entre EXEHDAnodos.

4.6.1 Collector

O serviço *Collector* é responsável pela extração da informação bruta (diretamente dos recursos envolvidos) que, posteriormente refinada, dará origem aos elementos de contexto. Para isto, o serviço *Collector* aglutina a informação oriunda de vários componentes monitores (*Monitor*) e as repassa aos consumidores registrados (*MonitoringConsumer*). Um componente *Monitor* gerencia um conjunto de sensores parametrizáveis. Por outro lado, entre os consumidores da informação extraída pela monitoração estão os serviços *ContextManager* e *Scheduler*.

Observe-se que, como decorrência da organização distribuída do EXEHDA, o coletor de um EXEHDAnodo pode também, se necessário, registrar-se como consumidor perante o coletor de outro EXEHDAnodo.

Na arquitetura de monitoração definida para o EXEHDA, ilustrada na figura 4.21, cada sensor contribui com a extração de um valor (*sensored value*) que descreve um aspecto específico, estático ou dinâmico, do recurso sendo monitorado. O conjunto dos sensores existentes em um dado EXEHDAnodo, assim como os parâmetros suportados por cada um destes sensores, integra a informação de descrição daquele nodo, disponibilizada no serviço CIB.

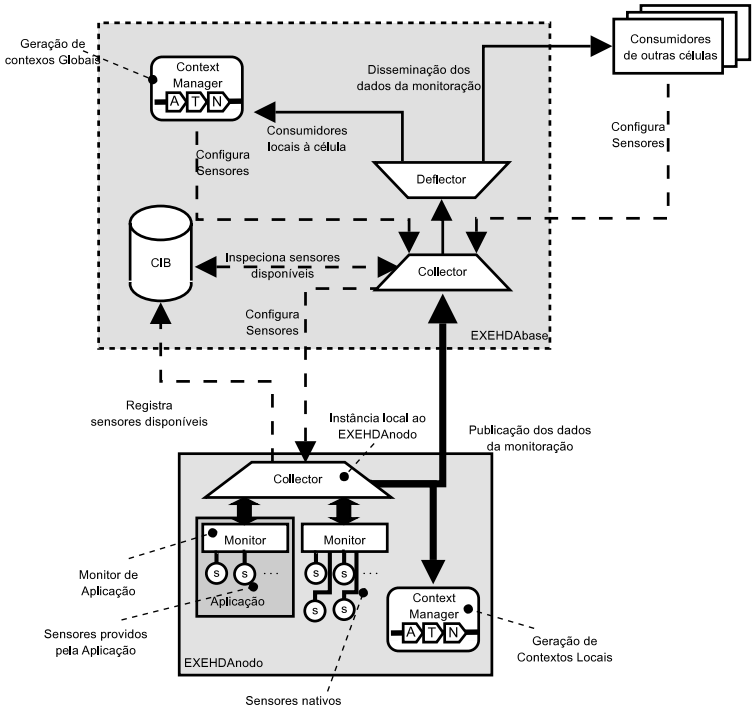


Figura 4.21: Arquitetura de monitoração

O serviço *Collector*, cuja interface é apresentada na figura 4.22, é responsável pela modificação dos parâmetros de operação dos sensores. A atuação do *Collector* ocorre por intermédio do *Monitor* associado ao sensor sendo re-parametrizado. É responsabilidade do *Collector* estabelecer parâmetros de operação que satisfaçam o caso mais exigente, considerando as necessidades dos consumidores registrados daquele sensor.

O registro e a remoção de um consumidor ocorrem por meio dos métodos `addConsumer` e `removeConsumer`, respectivamente. O registro de um consumidor resulta na geração de um `ConsumerId` para o mesmo, o qual pode ser posteriormente empregado em operações de reparametrização dos sensores. O `ConsumerId` gerado determina a forma pela qual a informação de monitoração será repassada para o consumidor em questão: (*i*) via o *callback* MonitorConsumer provido como parâmetro ou (*ii*) via canais de multicast providos pelo serviço *Deflector* (vide seção 4.6.2). Um consumidor pode induzir a adoção da estratégia de canais de multicast por parte do *Collector* fornecendo um *callback* MonitorConsumer nulo (*null*).

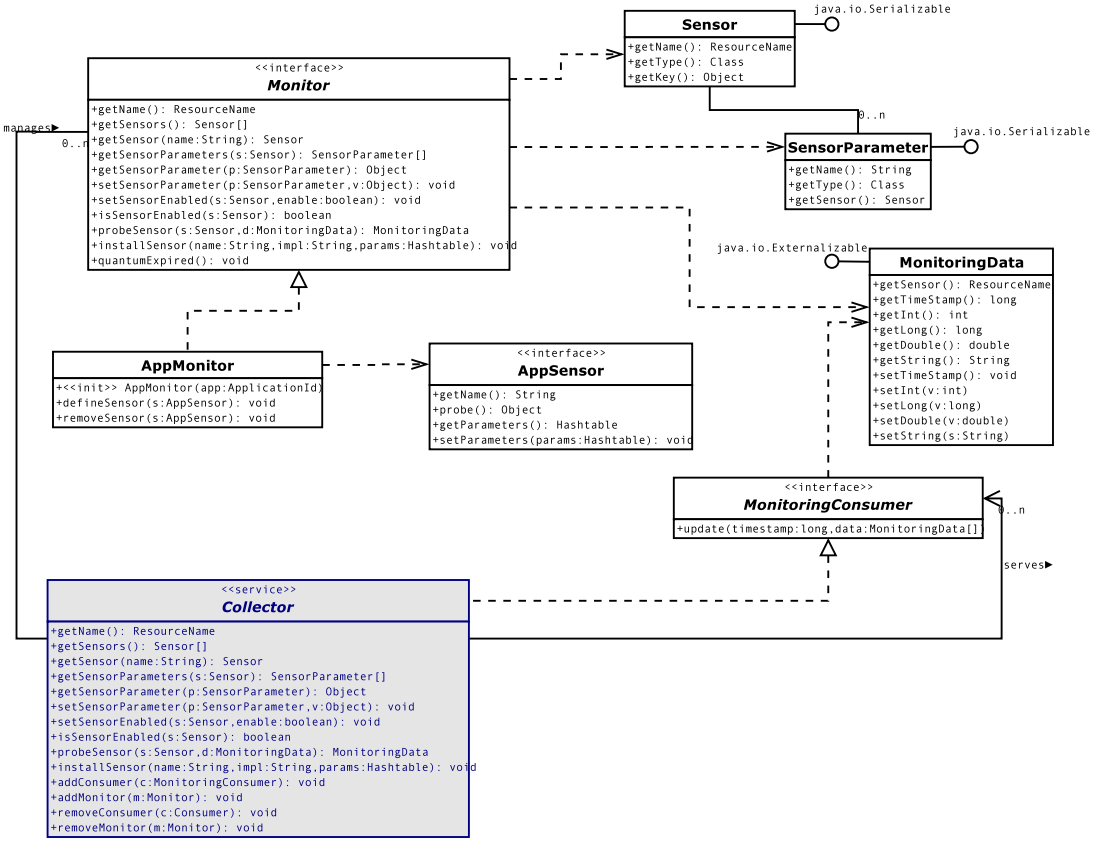


Figura 4.22: Diagrama de classes do serviço Collector

O controle da condição de publicação entre o *Collector* e os consumidores registrados é feito por meio do parâmetro *threashold*, que configura o limiar de variação do dado, de forma individualizada para cada sensor, acima do qual uma publicação do novo valor registrado pelo sensor é realizada.

A extração da informação junto aos monitores não ocorre em momentos arbitrários, mas apenas em instantes discretos, múltiplos de um determinado *quantum* de tempo. O *quantum* é um dos parâmetros de configuração do serviço *Collector* em cada

EXEHDAnodo, sendo utilizado para controlar o grau de intrusão do mecanismo de monitoração.

Transcorrido um quantum de tempo, o *Collector* notifica os monitores através do método `quantumExpired`, os quais então, executam uma operação de *polling* em cada um dos sensores ativos naquele momento no seu nodo. O critério de publicação (*treashold*) especificado para o sensor é aplicado, determinando a geração, ou não, de um evento para aquele sensor. Dessa forma, todos os eventos gerados dentro de um quantum são agrupados em uma única mensagem, reduzindo o tráfego de dados até os consumidores registrados.

4.6.2 Deflector

O objetivo do serviço *Deflector* é disponibilizar a abstração de canais de *multicast* para uso na disseminação das informações monitoradas. A presença do serviço *Deflector* é decorrência da busca de escalabilidade para a arquitetura de monitoração do EXEHDA.

A interface do serviço *Deflector* é apresentada na figura 4.23. O método `createChannel` é empregado para a criação de um canal de *multicast* e retorna um `ChannelId` para uso com o método `deliver`. Os métodos `joinChannel` e `leaveChannel` são empregados pelos consumidores da informação de *multicast* para inscrição e desinscrição, respectivamente, em um dado canal de *multicast*. Por sua vez, o método `deliver` dispara a disseminação de uma dada informação no canal de *multicast* especificado. Este frente de pesquisa do middleware está sendo aprofundada no trabalho [MOM 2004], em andamento.

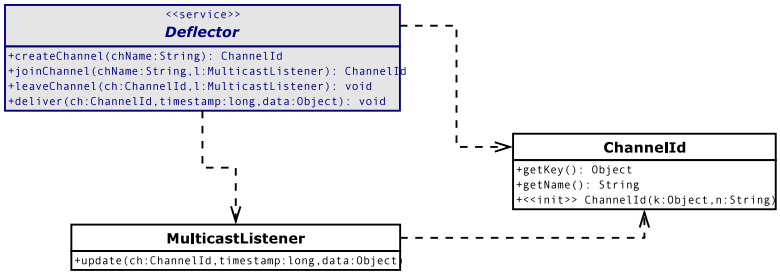


Figura 4.23: Diagrama de classes do serviço *Deflector*

O particionamento entre as funcionalidades de: (*i*) agregação de informações monitoradas, representada pelo serviço *Collector*, e de (*ii*) difusão de informações, relacionada ao serviço *Deflector*, é particularmente oportuno em se tratando da EXEHDAbase, por habilitar a distribuição destes serviços em equipamentos distintos, potencializando a escalabilidade do *middleware* como um todo.

4.6.3 ContextManager

O serviço *ContextManager*, cuja interface é apresentada na figura 4.24, é responsável pelo refinamento (tratamento) da informação bruta produzida pela monitoração para produção de informações abstratas referentes aos elementos de contexto (*contextualized value*).

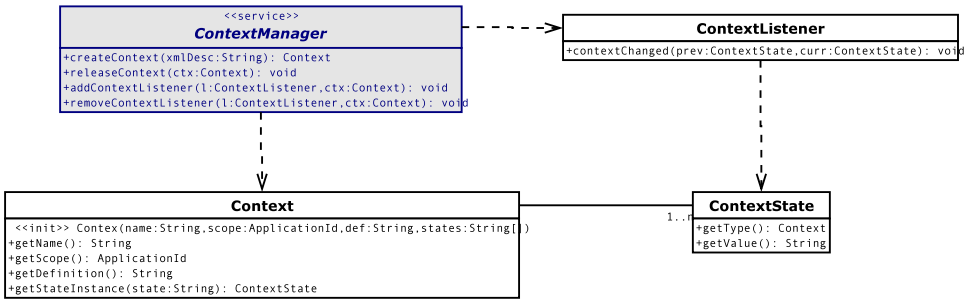


Figura 4.24: Diagrama de classes do serviço *ContextManager*

A definição de elementos de contexto ocorre por meio do método `createContext`, o qual retorna um objeto `Context` referente ao elemento de contexto criado e que define todos os estados (`ContextState`) possíveis para aquele elemento de contexto. O método `createContext` recebe como parâmetro uma descrição (XML) de como o dado referente àquele elemento de contexto deve ser produzido a partir da informação proveniente da monitoração. A figura 4.25 ilustra a definição de um elemento de contexto denominado “procpower”. Em contrapartida, o método `releaseContext` indica que um dado elemento de contexto não é mais necessário e os recursos alocados para a produção daquele elemento de contexto podem ser liberados. Após a definição de um dado contexto, os interessados no recebimento deste contexto podem registrar-se junto ao *ContextManager* via o método `addContextListener`. O cancelamento do registro de interesse em um determinado contexto ocorre pelo método `removeContextListener`

```
<CONTEXT n=”procpower”>
 <STATES>
   <STATE n=”low”/>
   <STATE n=”medium”/>
   <STATE n=”high”/>
   <STATE n=”unknown”/>
 <STATES>

 <INDEX>
   <SWITCH>
      <SENSOR n=”HOST_BENCH[bogomips]” scale=”10” />
      <SENSOR n="HOST_BENCH[linpack]" scale="4.75" />
      <COMPOSITE type="sum">
          <SENSOR n="FREE_PHYS_MEM" />
          <SENSOR n="HOST_BENCH[scimark]" scale="4.75" />
      </COMPOSITE>
   </SWITCH>
 </INDEX>

 <RANGES>
  <RANGE ub=”500” state=”low”/>
  <RANGE lb=”500” ub=”10000” state=”medium”/>
  <RANGE lb=”10000” state=”high”/>
  <DEFAULT state=”unknown”/>
 <RANGES>
</CONTEXT>
```

Figura 4.25: Exemplo de definição de contexto

A estratégia inicialmente prevista para a produção, em nível abstrato, da informação de contexto, consiste no emprego de cadeias de detecção de contexto, para cada contexto registrado, como ilustrado na figura 4.26. Estas cadeias são compostas por três elementos: (*i*) *aggregator*, (*ii*) *translator*, (*iii*) *notifier*. O agregador (*aggregator*) é

responsável pela composição dos dados de um ou mais sensores para a produção de valor dito agregado. O dado agregado é repassado ao componente tradutor para conversão do mesmo em um valor abstrato, conforme definido pelo programador da aplicação [AUG 2004]. O notificador (*notifier*), por sua vez, detecta alterações no dado abstrato gerado pelo tradutor (*translator*), gerando um evento de modificação de contexto.

Opcionalmente, a funcionalidade provida pelas cadeias de detecção de contexto pode ser estendida pela utilização de um componente que realize a predição (*Predictor*), em paralelo à filtragem produzida pelo agregador.

O serviço *ContextManager* está tendo suas funcionalidades revistas com as pesquisas referentes ao projeto contextS. Este projeto está sendo conduzido no escopo do Projeto ISAM, e conta com financiamento do CNPQ/FINEP/SEPIN [CTX 2003].

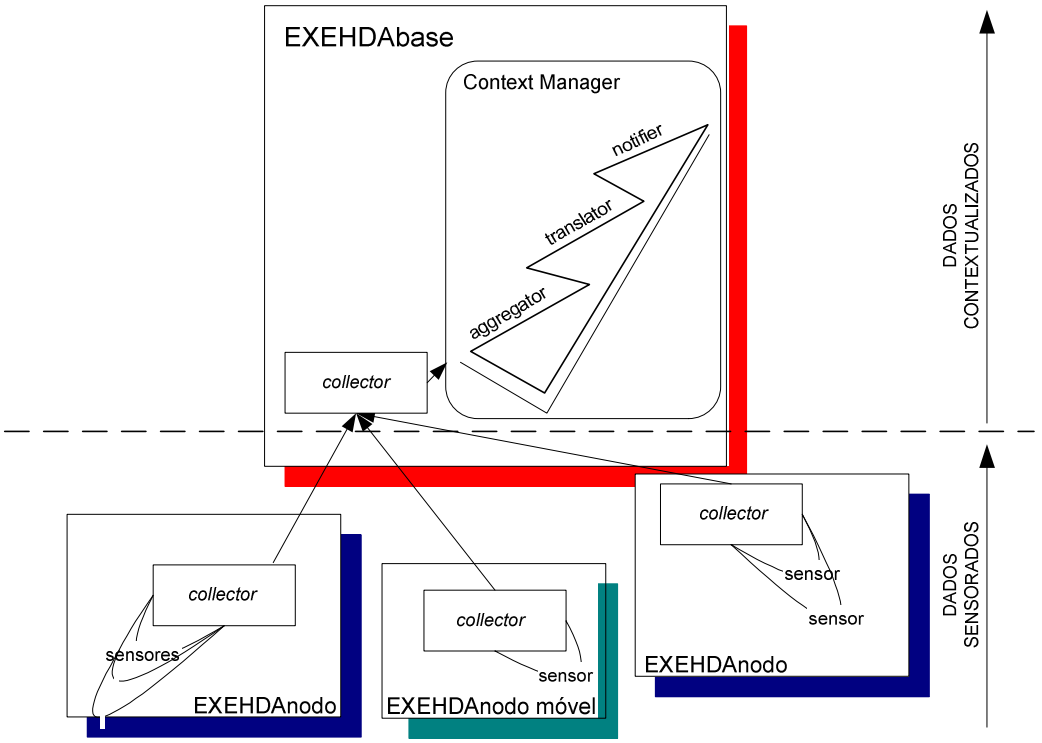


Figura 4.26: Construção da informação de contexto

4.6.4 AdaptEngine

No EXEHDA, o controle das adaptações de cunho funcional está a cargo do serviço *AdaptEngine*. Este serviço, cuja interface é apresentada na figura 4.27, provê facilidades para definição e gerência de comportamentos adaptativos por parte das aplicações.

Deste modo, libera o programador de gerenciar os aspectos de mais baixo nível envolvidos na definição e liberação dos elementos de contexto junto ao *ContextManager*.

O serviço *AdaptEngine* emprega uma representação baseada em strings para elementos de contexto e seus estados na forma “<elemento>:<estado>”. Este serviço procede, quando for necessária, a parametrização do serviço *ContextManager*, a partir das definições providas através do arquivo `context.xml`, para sintetização dos

elementos de contexto de interesse de cada aplicação e registra seu interesse em receber notificações de alterações no estado daqueles elementos de contexto. Face à notificação de alteração no estado de algum elemento de contexto, o serviço *AdaptEngine* é responsável pela gerência e notificação dos componentes registrados como adaptativos/sensíveis àquele elemento de contexto. Nesse sentido, o registro de tais componentes dá-se pelo método `addContextListener` e o cancelamento de um registro prévio pelo método `removeContextListener`.

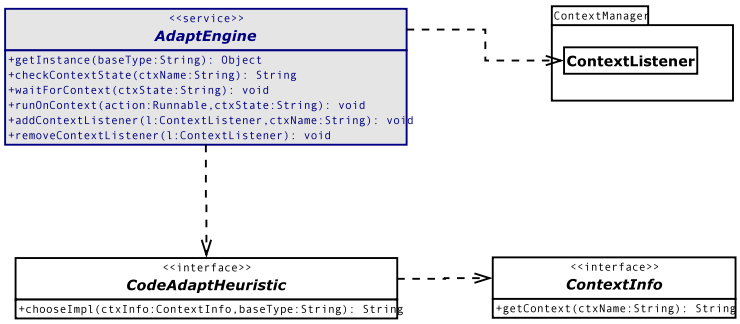


Figura 4.27: Diagrama de classes do serviço *AdaptEngine*

Outra função do serviço *AdaptEngine* é prover um mecanismo de carga de código contextualizado. Desta forma, com base na informação do estado do elemento de contexto fornecido pelo serviço *ContextManager*, seleciona e carrega o código correspondente dentre alternativas de código definidas na política de adaptação funcional vigente. Esta operação de carga adaptativa ocorre através do método `getInstance`.

Este serviço disponibiliza ainda métodos para a ativação de ações quando determinado estado de um elemento de contexto tornar-se ativo. Nesse sentido, o método `waitForContext` permite aguardar pela ocorrência de um determinado estado de contexto, enquanto que o método `runOnContext` permite agendar uma ação a ser executada assincronamente com o fluxo principal de execução quando o estado de contexto especificado tornar-se ativo.

Um elemento chave no mecanismo de adaptação funcional é a heurística de adaptação (CodeAdaptHeuristc). Esta heurística é personalizada por aplicação, sendo consultada pelo serviço *AdaptEngine*, quando da ativação do método `getInstance`, para seleção da implementação mais oportuna de um dado componente considerando a situação atual do contexto. A heurística a ser usada por uma dada aplicação é determinada por um atributo definido no descritor de disparo da aplicação.

Caso a aplicação não defina explicitamente uma heurística de adaptação funcional, a implementação padrão é assumida. Esta interpreta, em tempo de execução, a política de adaptação funcional definida através do arquivo `adapters.xml` distribuído com a aplicação. A figura 4.28 ilustra o formato do arquivo `adapters.xml`.

Entre as possíveis razões para uma aplicação modificar a heurística padrão estão: (*i*) a modificação da sintaxe empregada no `adapters.xml` ou (*ii*) o desempenho. No primeiro caso, entende-se que a definição dos adaptadores possa ser ajustada para uso de termos mais significativos no escopo da linguagem ou paradigma de programação empregados no desenvolvimento da aplicação. Por outro lado, a melhoria de desempenho viria da possibilidade da informação contida na política de adaptação funcional ser compilada e incorporada à própria heurística evitando, neste caso, a

necessidade de interpretar-se o `adapters.xml` em tempo de execução. Soluções mistas, representando um ajuste entre sintaxe e desempenho, também são possíveis.

```
<ADAPTERS>
   <ADAPTER baseType="" context="">
      <CTXSTATE v="ctx1" impl="impl1"/>
      <CTXSTATE v="ctx2" impl="impl2"/>
      <CTXSTATE v="ctx3" impl="impl3"/>
      <DEFAULT impl="impl4">
   </ADAPTER>
   ...
</ADAPTERS>
```

Figura 4.28: Exemplo de política de adaptação funcional

4.6.5 Scheduler

O serviço *Scheduler* é o serviço central na gerência das adaptações de cunho não-funcional no EXEHDA, isto é, que não implicam alteração de código. Nesse sentido, o *Scheduler* emprega a informação de monitoração, obtida junto ao serviço *Collector*, para orientar operações de mapeamento. Essas operações decorrem de instanciações remotas ou migrações realizadas pelo serviço *Executor*, ou quando de chamadas de re-escalonamento, originadas do estado atual de um recurso não satisfazer mais as necessidades de um objeto anteriormente a ele alocado. A interface do serviço *Scheduler* é apresentada na figura 4.29.

Os dois primeiros métodos, `chooseCreationHost` e `chooseMigrationHost`, são acionados, tipicamente, pela heurística de escalonamento da aplicação (*SchedHeuristic*), a qual é consultada pelo serviço *Executor* para a seleção do EXEHDAnodo que deverá receber um OX sendo instanciado ou migrado respectivamente. Estes métodos recebem como parâmetro um conjunto de restrições (*SchedConstraint*) que devem ser consideradas quando da seleção do nodo. Estas restrições podem assumir diversas naturezas: (*i*) maximização ou minimização de algum índice global/celular do sistema, (*ii*) posicionamento relativo a outros componentes já existentes, (*iii*) características do EXEHDAnodo e modo (*iv*) modo de alocação.

O terceiro método, `reschedule`, estabelece um novo conjunto de restrições que devem ser satisfeitas pelo EXEHDAnodo que hospeda um determinado OX. A não satisfação das novas restrições definidas determina a migração daquele OX para um outro EXEHDAnodo que preencha os requisitos. Nesta perspectiva, o *Scheduler* pode proceder à verificação das restrições uma única vez, por ocasião da invocação do reschedule, ou periodicamente, de forma automática. A seleção entre estes comportamentos dà-se pelo terceiro parâmetro do método `reschedule`. No caso do re-escalonamento automático, a política de escalonamento (restrições) associada ao OX é armazenada no atributo 'sched-policy' daquele OX, o qual é gerenciado através do serviço *OXManager*.

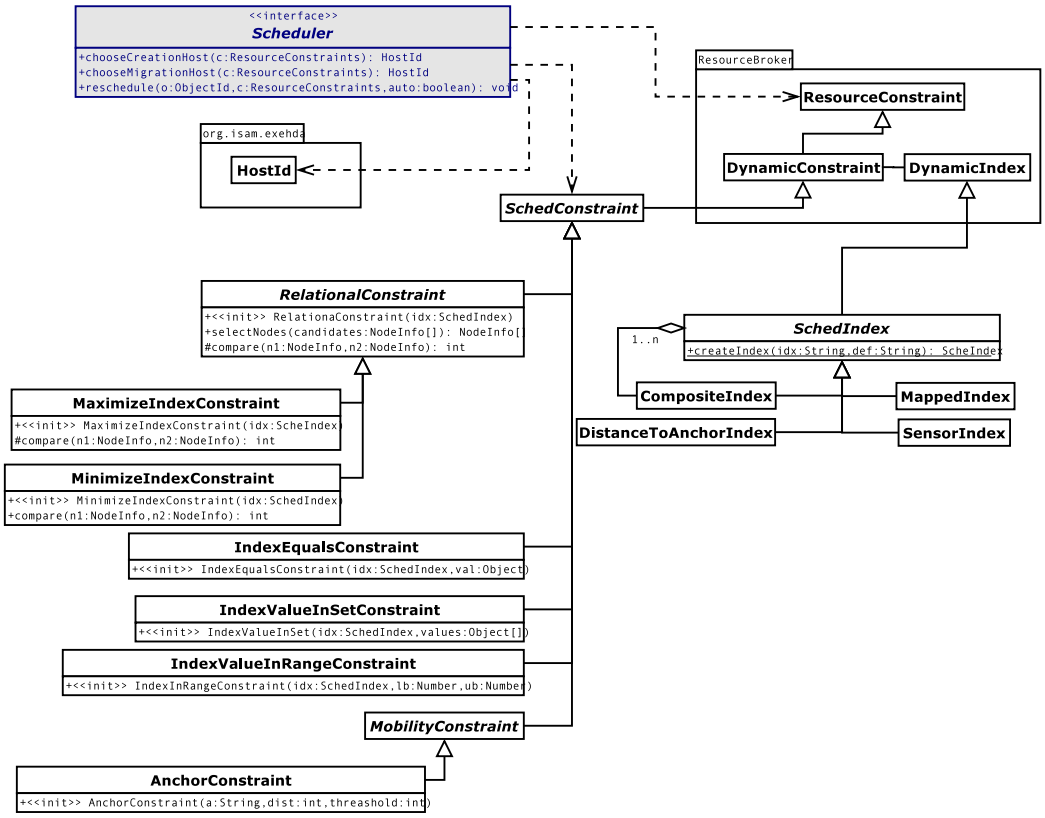


Figura 4.29: Diagrama de classes do serviço *Scheduler*

4.7 Subsistema de comunicação

A natureza da mobilidade do hardware e, na maioria das vezes, também a do software, não garante a interação contínua entre os componentes da aplicação distribuída. As desconexões são comuns, não somente devido à existência de alguns *links* sem fio, mas sobretudo como uma estratégia para economia de energia nos dispositivos móveis [ALB 2000]. O subsistema de comunicação do EXEHDA disponibiliza mecanismos que atendem estes aspectos da Computação Pervasiva. Integram este subsistema os serviços *Dispatcher*, *WORB*, *CCManager*, os quais contemplam modelos com níveis diferenciados de abstração para as comunicações.

4.7.1 Dispatcher

O serviço *Dispatcher* disponibiliza o modelo de comunicação mais elementar do EXEHDA: troca de mensagens ponto-a-ponto com garantia de entrega e ordenamento das mensagens, o qual é especializado para operação no ambiente pervasivo. As mensagens trafegam entre instâncias do *Dispatcher* localizadas em nodos diferentes através de estruturas denominadas canais. Nesse sentido, quando de sua incialização, o *Dispatcher* atualiza a informação do EXEHDAnodo na CIB, em específico o atributo *contactAddress*, provendo uma lista de protocolos e endereços que podem ser utilizados para alcançar aquele EXEHDAnodo.

A interface do serviço *Dispatcher* pode ser vista na figura 4.30. Essa interface contempla métodos para alocação e liberação de portas e buffers (métodos `createPort`, `releasePort`, `allocBuffer` e `releaseBuffer`), além de operações

para envio e recepção de mensagens (métodos `send` e `receive`) sobre portas previamente alocadas. Adicionalmente, considerando o aspecto migração de componentes de software usual na Computação Pervasiva, são disponibilizados métodos para a suspensão, reativação e redirecionamento de portas (métodos `suspendPort`, `resumePort` e `redirectPort`, respectivamente). Do ponto de vista do emissor da mensagem, a migração de uma porta destino para outro EXEHDAnodo é transparente, sendo as mensagens destinadas à localização antiga, automaticamente redirecionadas para a nova localização da porta.

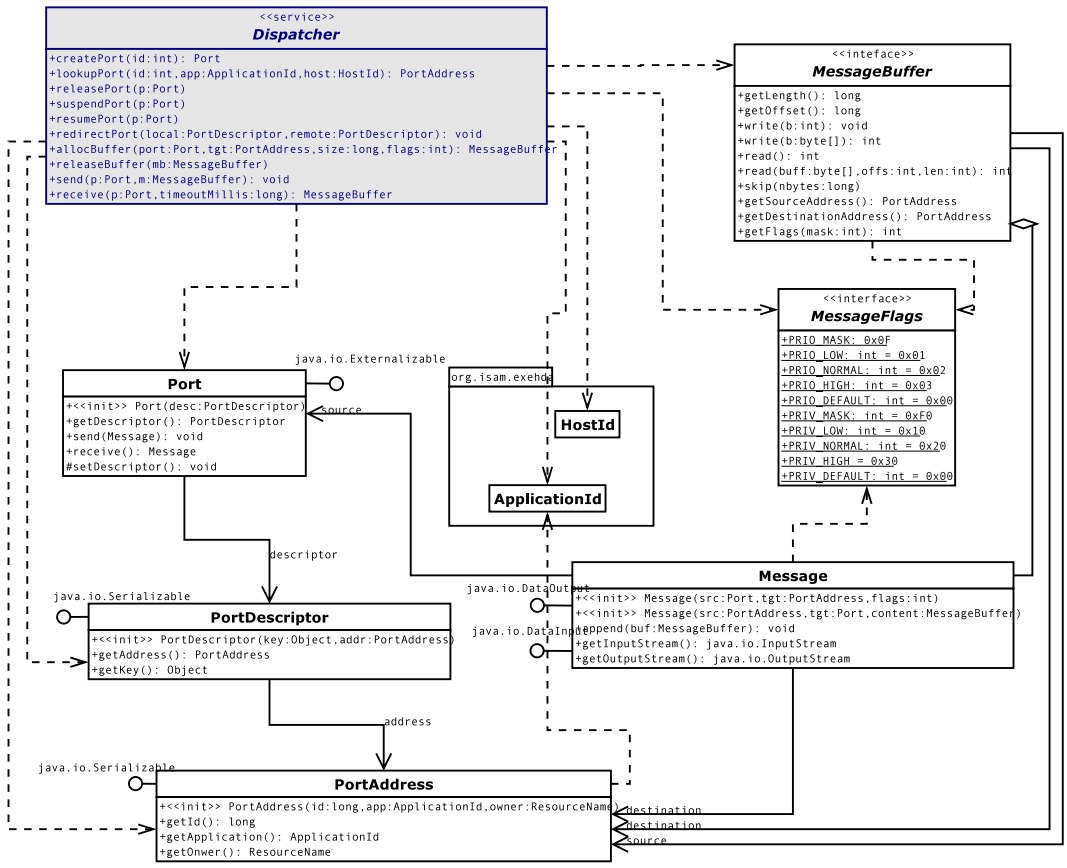


Figura 4.30: Diagrama de classes do serviço *Dispatcher*

Embora não diretamente visível na interface do serviço *Dispatcher*, a abstração canal é central na definição e entendimento da semântica das operações definidas para este serviço, sendo sinônimo das propriedades de garantia da entrega e ordenamento das mensagens no EXEHDA.

Os canais são abstrações ponto-a-ponto, estabelecidas entre EXEHDAnodos que necessitam comunicação, sendo criados no escopo de uma aplicação específica, ou seja, são estruturas de uso privado da aplicação. A alocação e a liberação de canais ocorre de forma implícita em função das operações de envio e recepção de mensagens. O protocolo de estabelecimento de um canal compreende, além da autenticação dos EXEHDAnodos envolvidos, a identificação da aplicação à qual aquele canal se destina.

De maneira a habilitar otimizações nas camadas inferiores associadas à implementação da comunicação (i.e., sistema operacional ou APIs de alto desempenho), o modelo de comunicação define chamadas específicas para a alocação dos buffers que serão empregados na composição das mensagens. Nesse sentido, além do destino da

mensagem, na alocação de um buffer (`MessageBuffer`) é definido o nível de privacidade requerido por aquela mensagem. Com base nessas informações, a implementação do *Dispatcher* seleciona a tecnologia de comunicação a ser utilizado pelo canal empregado para conexão com a instância do *Dispatcher* em execução no EXEHDAnodo destino. Com base na tecnologia selecionada, o *Dispatcher* procede, então, a alocação do buffer adequado à mesma. Os primeiros esforços de pesquisa nesta direção fizeram parte do trabalho [SIL 2003].

No que se refere à manutenção da consistência das comunicações durante os períodos de desconexão planejada, o *Dispatcher* emprega internamente um mecanismo de *checkpointing* do estado dos canais, provendo um tratamento transparente da desconexão planejada ao utilizador da abstração canal. Durante o período de desconexão as mensagens geradas pela aplicação são armazenadas no canal, sendo transmitidas quando da reconexão. Registre-se que o tamanho da fila de armazenamento utilizado é um parâmetro da implementação.

4.7.2 WORB

Com o objetivo de simplificar a construção de serviços distribuídos, o EXEHDA disponibiliza o serviço *WORB*, permitindo que os programadores focalizem esforços no refinamento da semântica distribuída associada ao serviço em desenvolvimento, abstraindo aspectos de baixo nível relativos ao tratamento das comunicações em rede. Para tanto, oferece um modelo de comunicação baseado em invocações remotas de método, similar ao RMI, porém sem exigir a manutenção da conexão durante toda a execução da chamada remota.

A funcionalidade do serviço *WORB* é construída sobre a provida pelo serviço *Dispatcher*, herdando, portanto, a característica de escopo individualizado por aplicação, assim como um tratamento básico da desconexão planejada. Adicionalmente, o *WORB* emprega um protocolo de invocação remota de métodos também sintonizado à premissa da desconexão, de forma a atingir uma melhor eficiência nas comunicações. Por este protocolo, cada chamada remota recebe um identificador único, o qual permanece válido durante uma eventual desconexão, sendo utilizado quando da reconexão para recuperação do resultado da invocação remota.

A interface do serviço WORB, ilustrada na figura 4.31, disponibiliza métodos para o registro (`exportService`) e cancelamento do registro (`unexportService`) de um objeto (serviço na perspectiva do WORB), respectivamente habilitando e cancelando o suporte a que este objeto receba invocações remotas de método. Adicionalmente, disponibiliza o método `lookupService`, o qual possibilita a obtenção de uma referência para um dado serviço remoto, e `setExceptionHandler` que permite modificar o tratador de exceções de comunicação associado a uma dada referência remota. Por sua vez, o método `getClientHost` quando utilizado dentro de um método previamente exportado pelo WORB, permite identificar o EXEHDAnodo originador da invocação remota sendo tratada.

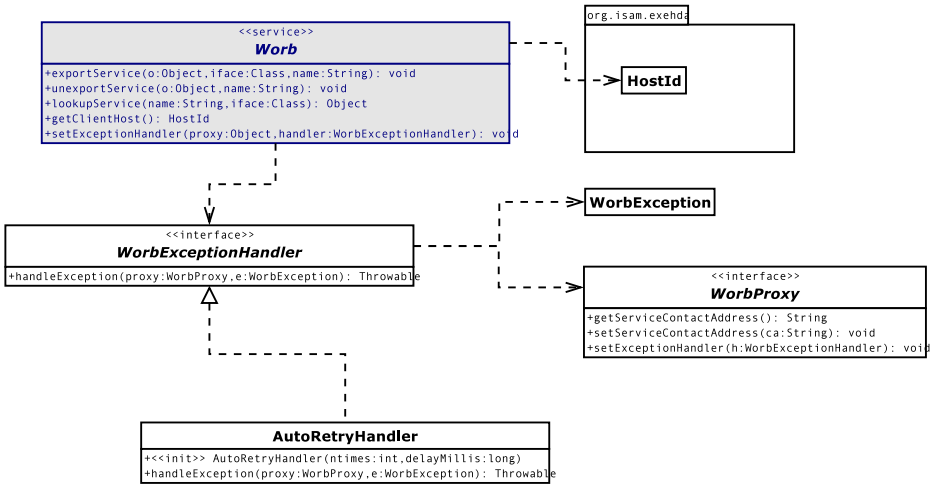


Figura 4.31: Diagrama de classes do serviço WORB

4.7.3 CCManager

Considerando a premissa da mobilidade lógica dos componentes que constituem as aplicações pervasivas, o suporte a um mecanismo de comunicação com característica de desacoplamento temporal e espacial é particularmente oportuno, à medida que este simplifica a construção de tais aplicações.

O serviço *CCManager* vem atender a esta demanda, disponibilizando um mecanismo baseado na abstração espaço de tuplas [GEL 85], o qual prescinde da coexistência temporal de emissor e receptor. Outro aspecto oportuno desta abstração é a facilidade com que podem ser implementados outros padrões de comunicação, além do ponto-a-ponto.

A interface do serviço *CCManager* do EXEHDA é apresentada na figura 4.32. Nela são disponibilizados métodos para criação e destruição de espaços de tuplas (`createSpace` e `destroySpace`), obtenção de liberação de referência para um espaço de tuplas existente (`joinSpace` e `leaveSpace`), assim como métodos para inserção e consumo de tuplas (`in` e `out`).

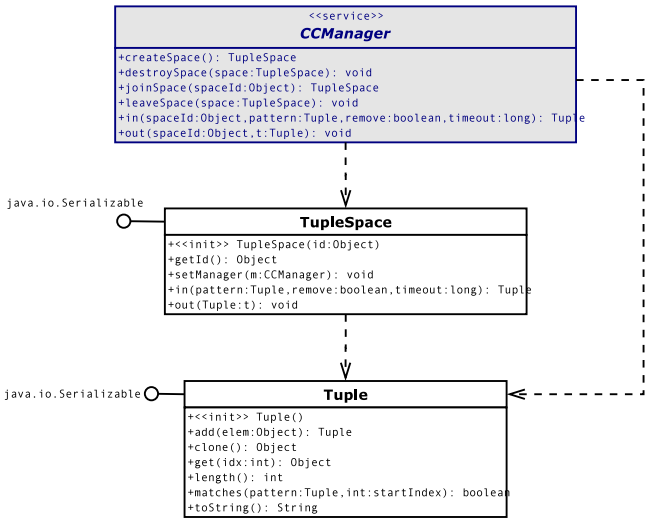


Figura 4.32: Diagrama de classes do serviço *CCManager*

Como estratégia para conferir escalabilidade à solução, os espaços de tuplas são inicialmente criados locais ao EXEHDAnodo. Na medida em que requisições de `joinSpace` são emitidas em outros EXEHDAnodos, o espaço de tuplas é dinamicamente expandido de forma a tornar-se acessível também àquele EXEHDAnodo. Este procedimento de expansão dinâmica envolve a instância celular do serviço *CCManager*.

4.8 Subsistema de acesso pervasivo

A premissa de acesso em qualquer lugar, todo o tempo, a dados e código da Computação Pervasiva, requer um suporte do *middleware*. Os serviços que compõem este subsistema no EXEHDA são: BDA, AVU, *SessionManager* e *Gatekeeper*.

4.8.1 BDA

O ambiente de Computação Pervasiva tem por premissas (*i*) a possibilidade de o usuário disparar aplicações a partir de qualquer nodo integrante do sistema e, após o disparo, (*ii*) a mobilidade parcial ou integral de tais aplicações em resposta a modificações em seu contexto de execução, como, por exemplo, a movimentação do usuário ou alteração na condição de carga dos dispositivos atualmente em uso pela aplicação.

Para tanto, a capacidade de instalação de código sob demanda é uma necessidade inerente à execução de aplicações na Computação Pervasiva. Seria impraticável manter todo o universo de software disponível instalado e atualizado (com coerência de versões) em todos os dispositivos do sistema.

Por sua vez, a implementação do mecanismo de instalação sob demanda requer a existência de um repositório de código que forneça a mesma visão do software disponibilizado a partir de qualquer dispositivo do ISAMpe, mesmo após migrações.

Outras funcionalidades para este repositório pervasivo também são desejáveis. Considerando a perspectiva de que as diversas aplicações disponibilizadas sigam linhas de evolução independentes, o suporte a controle de versões é oportuno na direção da manutenção da operacionalidade das diferentes aplicações.

O serviço BDA (Base de Dados pervasiva das Aplicações) do EXEHDA tem sua interface apresentada na figura 4.33. Ele contempla métodos para a recuperação do código integral de uma aplicação ou de componentes específicos e suas dependências, operações estas disponibilizadas através das duas variantes do método `fetch`. Por outro lado, este serviço inclui ainda métodos para a gerência das aplicações instaladas (`install`, `remove`, `get/setDescriptor`, `list` e `listCathegories`). Nesse sentido, o controle de versões anteriormente mencionado pode ser obtido representando-se a versão de interesse na própria URL utilizada no acesso ao serviço BDA (e.g., “bda://ufrgs/computer-graphics/mandelbrot.jar?version=1.3”).

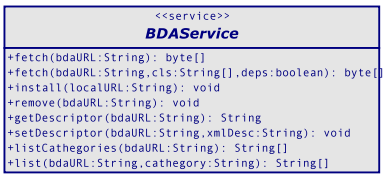


Figura 4.33: Diagrama de classes do serviço BDA

O padrão de utilização concebido para o serviço BDA define que as requisições geradas nos EXEHDAnodos de uma determinada célula são sempre direcionadas à instância celular do serviço BDA da mesma EXEHDAcel onde foi gerada a requisição. Esta, se necessário, realiza o acesso a instâncias remotas (BDAs de outras células) para satisfazer uma dada requisição. Desta forma, a instância celular fica habilitada a empregar mecanismos de *caching* para o código das aplicações buscando melhoria no desempenho, ao mesmo tempo em que preserva a visão de acesso pervasivo para o cliente do serviço BDA.

4.8.2 AVU

A premissa *siga-me* definida no ISAM reflete-se não só nas aplicações que um usuário atualmente executa, mas no seu ambiente computacional como um todo. Este engloba, além das aplicações em execução, as informações de personalização das aplicações definidas pelo usuário, o conjunto de aplicações instaladas (i.e. passíveis de serem disparadas por aquele usuário), como também seus arquivos privados. É atribuição do serviço AVU (Ambiente Virtual do Usuário) do EXEHDA a manutenção do acesso pervasivo a este ambiente virtual, da forma mais eficiente possível.

Com este objetivo, o serviço AVU adota uma estratégia de utilização análoga à empregada no serviço BDA. As requisições geradas pela instância local ao EXEHDAnodo do serviço AVU são direcionadas à instância celular do mesmo serviço em execução na célula local.

A instância celular, por sua vez, consulta à célula “home” do usuário de forma a descobrir a ultima localização física de seu ambiente virtual. De posse desta informação, este pode acessar o serviço AVU remoto para conclusão do tratamento da requisição. Quando oportuno, considerando o estado do link de rede, a instância do serviço AVU em execução na célula local pode optar por mover integralmente o ambiente virtual do usuário para a célula local, de forma a otimizar acessos futuros. Opta-se no EXEHDA pela movimentação, e não replicação, com vistas a minimizar os problemas associados à manutenção das réplicas do ambiente virtual. Considerando a natureza individualizada, por usuário, do ambiente virtual, a movimentação é uma abordagem adequada para a satisfação da semântica *siga-me*.

A interface do serviço AVU é apresentada na figura 4.34. Este serviço define operações tradicionais como abertura e fechamento de arquivos (`open` e `close`), leitura e escrita (`read` e `write`), reposicionamento do ponteiro de arquivo (`seek`), listagem de diretórios (`list`), movimentação e remoção de arquivos dentro do AVU (`move` e `remove`). Adicionalmente, inclui operações `prefetch` e `release`, direcionadas à otimização do acesso aos dados armazenados no AVU face aos aspectos de mobilidade e desconexão planejada.

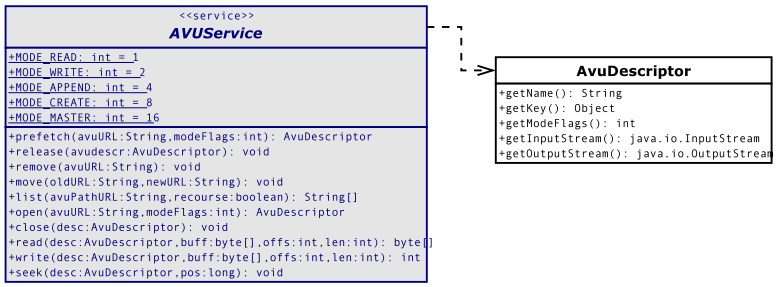


Figura 4.34: Diagrama de classes do serviço AVU

Nesse sentido, a operação `prefetch` indica ao sistema que determinado arquivo no AVU será em breve utilizado, permitindo, por exemplo, que o serviço AVU armazene previamente aquele arquivo no dispositivo local, habilitando a continuação da aplicação mesmo que uma desconexão planejada venha a ocorrer. Em contrapartida, a operação `release` indica que um determinado arquivo não será utilizado em um futuro próximo e, portanto, o sistema pode removê-lo da cache local se oportuno (por exemplo, para liberar memória para outra operação). Observe-se que as operações `prefetch` e `release` consistem em otimizações adicionais, não sendo estritamente necessárias para o acesso a um arquivo armazenado no AVU.

4.8.3 SessionManager

O serviço *SessionManager* do EXEHDA é responsável pela gerência da sessão de trabalho do usuário, sendo definida pelo conjunto de aplicações correntemente em execução para aquele usuário. A informação que descreve o estado da sessão de trabalho é armazenada no AVU, estando portando disponível de forma pervasiva. A figura 4.35 ilustra a interface do serviço *SessionManager*.

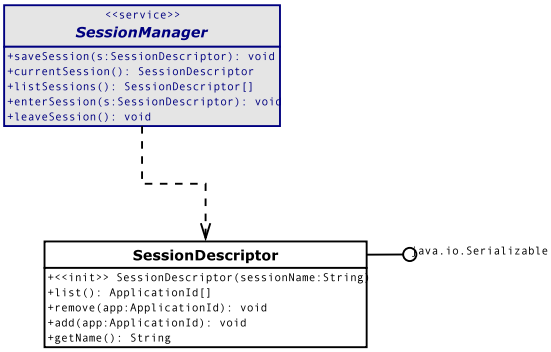


Figura 4.35: Diagrama de classes do serviço *SessionManager*

Na perspectiva do serviço *SessionManager*, uma sessão é representada por um objeto SessionDescriptor, o qual define métodos para inclusão e remoção de aplicações. O *SessionManager*, por sua vez, disponibiliza métodos para salvamento (`saveSession`) e carga de uma determinada sessão (`enterSession`), além de métodos para listar as sessões definidas (`listSessions`) e obter ou abandonar a sessão corrente (`currentSession e leaveSession`).

Tipicamente, o serviço *SessionManager* é ativado através do serviço *Gatekeeper*, após a conclusão com sucesso do procedimento de autenticação do usuário. Como resultado, o *SessionManager* recupera o estado da sessão *default* do usuário, restaurando

a execução das aplicações correspondentes (vide seção 6.2.1). Adicionalmente, a funcionalidade do *SessionManager* pode estar acessível através de aplicações de usuário, como o ISAMDesktop, permitindo assim um controle mais preciso das aplicações contidas em uma dada sessão, não ficando restrito à manipulação da sessão *default*.

4.8.4 Gatekeeper

O serviço *Gatekeeper* é responsável por intermediar acessos entre as entidades externas à plataforma ISAM e os serviços do *middleware* de execução, conduzindo os procedimentos de autenticação necessários. A interface do serviço *Gatekeeper* é apresentada na figura 4.36.

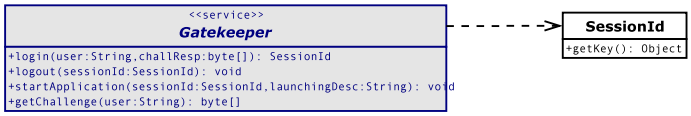


Figura 4.36: Diagrama de classes do serviço *Gatekeeper*

O protocolo de autenticação do usuário empregado no EXEHDA baseia-se em um mecanismo de chave pública/privada [MEN 97] e envolve inicialmente a obtenção de um desafio (*challenge*) junto ao *Gatekeeper*, o que ocorre através do método `getChallenge`. O desafio consiste de um número inteiro longo, o qual é criptografado com a chave pública do usuário, previamente disponibilizada na CIB. A ferramenta em uso pelo usuário deve acessar a chave privada do usuário armazenada em um meio portátil (exemplo mini-cd, disquete, smart-card, memory-key, etc.), decriptografar o desafio, incrementá-lo e voltar a criptografá-lo. Este novo valor é retornado ao *Gatekeeper*, via o método `login`, o qual o decriptografa utilizando novamente a chave pública armazenada na CIB, e verifica se o valor retornado é igual ao valor original acrescido de 1.

Em caso de sucesso, um identificador de sessão é retornado ao usuário, permitindo que este execute o disparo de aplicações a partir do método `startApplication`. O método logout permite invalidar um identificador de sessão previamente alocado. Observe-se que a *SessionId*, por aspectos de segurança, expira automaticamente após determinado tempo, mesmo que o procedimento de *logout* não tendo sido efetuado.

O próximo capítulo trata de como os serviços do EXEHDA são utilizados na adaptação de aplicações ISAMadapt. Para tal, é feita uma associação com as principais abstrações da linguagem e os mecanismos disponibilizados pelo *middleware*.

5 O SUPORTE DO EXEHDA ÀS APLICAÇÕES ISAMADAPT

Acorns were good until bread was found.

- Francis Bacon

A linguagem ISAMadapt foi concebida para o desenvolvimento de aplicações direcionadas para Computação Pervasiva, e seu escopo está comprometido com as premissas da arquitetura ISAM. A mesma instancia os conceitos de consciência do contexto e adaptação dinâmica, através de abstrações que compreendem definição de contexto, comportamentos alternativos, comandos e políticas de adaptação [AUG 2004].

As aplicações ISAMadapt executam sobre o EXEHDA, o qual fornece um ambiente de execução pervasivo para estas. O objetivo deste capítulo é apresentar do ponto de vista do EXEHDA, o suporte que este oferece às abstrações definidas na linguagem ISAMadapt.

5.1 ISAMadapt: modelo de programação

A aplicação ISAMadapt é modelada considerando a estrutura organizacional do Holoparadigma [BAR 2002]. Segundo a proposta Holoparadigma, a aplicação é modelada com entes (entidades de existência e mobilidade) e símbolos (entidades de informação). Cada ente define um comportamento funcional (código imperativo e/ou lógico), seu estado (variáveis e/ou fatos) e sua história. Um ente pode ser composto de outros (figura 5.1). Esta organização hierárquica decorre do processo de criação de entes, chamado clonagem, ou de migração de entes: um ente quando *clona* outro, cria um filho; um ente quando migra para dentro de outro, torna-se filho deste.

A comunicação e a sincronização, entre os entes, é realizada através da história. A história é baseada no modelo de *blackboards*. O modelo Holoparadigma define que um ente somente tem acesso a sua própria história e à história do seu ente pai. Para a comunicação com outro ente, o ente pode se deslocar para dentro do mesmo, tornando-se filho deste e, a partir daí, acessar a história do pai (mobilidade lógica). A mobilidade física é implícita, e seu gerenciamento é responsabilidade do sistema de execução. A abstração história é suportada no EXEHDA através de espaços de tuplas de objetos distribuídos (vide seção 4.7.3).

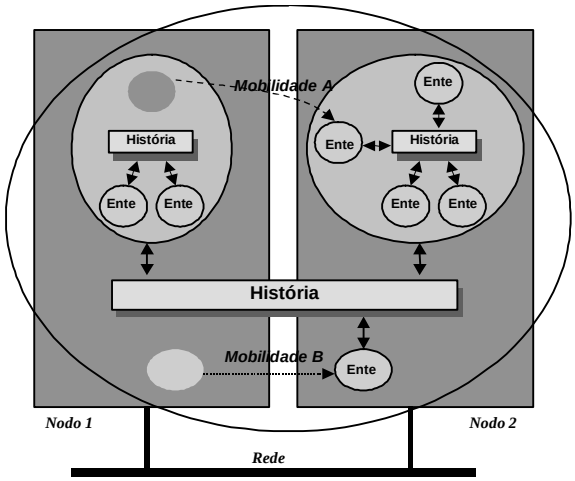


Figura 5.1: Estrutura de entes no HoloParadigma

A linguagem ISAMadapt estende a proposta Holoparadigma pela inclusão de construções para expressão do comportamento adaptativo. As abstrações disponibilizadas na linguagem ISAMadapt são relativas à consciência de contexto e à adaptação dinâmica, e compreendem: contexto, comportamentos alternativos, comandos e políticas de adaptação. Uma aplicação ISAMadapt é estruturada com quatro componentes:

1. elementos de contexto de interesse da aplicação;
2. entes/métodos funcionais e adaptativos, que implementam a lógica da aplicação e o comportamento adaptativo. Os entes e métodos podem fazer uso de comandos de adaptação, que disponibilizam facilidades para expressão do comportamento consciente do contexto, tais como migração e descoberta de recursos;
3. adaptadores, que implementam os códigos alternativos dos entes/métodos adaptativos correspondentes ao estado do elemento de contexto corrente;
4. políticas de adaptação, que orientam o sistema de execução na sua tomada de decisão.

Na fase de desenvolvimento, o programador dispõe de um ambiente de programação que o orienta na descrição da aplicação e do comportamento adaptativo desta. O programador ISAMadapt codifica os aspectos da adaptação (quando necessários): define os elementos de contexto de interesse da aplicação, codifica comportamentos alternativos ajustados às condições ambientais e seleciona as políticas de adaptação a serem disponibilizadas ao EXEHDA. Esta organização para o projeto da aplicação está em conformidade com uma metodologia (em desenvolvimento) para a criação de aplicações conscientes do contexto [REI 2002].

5.2 Integração ISAMadapt e EXEHDA

A integração da linguagem ISAMadapt com o EXEHDA acontece de duas maneiras: explicitamente através de chamadas aos serviços do *middleware*, ou implicitamente via arquivos de políticas descritos em XML. Os arquivos de políticas parametrizam a operação de serviços do EXEHDA. Uma visão geral desta integração está apresentada na figura 5.2.

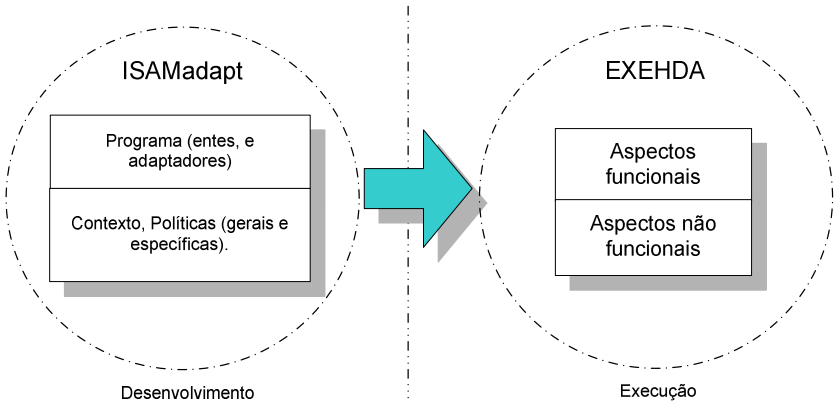


Figura 5.2: Visão geral da integração EXEHDA e ISAMadapt

O suporte para as abstrações do ISAMadapt provido pelo EXEHDA será apresentado nas seções a seguir.

5.2.1 O Serviço BeingManager e o suporte à abstração História

O ente pode operar sobre a História utilizando operações definidas pela Hololinguagem [BAR 2002] as quais contemplam a escrita, além de versões bloqueantes e não-bloqueantes de operações de leitura, com e sem remoção da tupla correspondente da História. Abaixo a sintaxe destes comandos:

- escrita: history ! <tupla>
- leitura bloqueante: history # <gabarito>
- leitura não-bloqueante: history #? <gabarito>
- retirada bloqueante: history . <gabarito>
- retirada não-bloqueante: history .? <gabarito>

O serviço *BeingManager*, cuja interface é apresentada na figura 5.3, dispobiliza métodos para a criação de novos entes (`createBeing`), assim como para movimentação na hieraquia lógica da aplicação. Adicionalmente, define métodos que permitem recuperar a identificação do ente (`getName`) ou de seu pai (`getParentName`), assim como referências para a sua história (`getHistory`) ou para a história de seu pai (`getParentHistory`). É possivel ainda enumerar a lista de filhos atuais do próprio ente (`getChildren`).

A funcionalidade do serviço *BeingManager* é construída sobre a provida pelo *OXManager*. Em específico, o *BeingManager* emprega os seguintes atributos do OX para armazenamento de informação pertinente a gerência do ente:

- being.name: nome do ente
- being.parent: ObjectId do OX correspondente ao pai atual do ente.
- being.history: identificador do espaço de tuplas que correspondente a história do ente.

Observe-se que a implementação da mobilidade lógica consiste na atualização do atributo 'being.parent'. As demais operações podem sempre se deduzidas a partir do identificador do OX associado a um determinado ente.

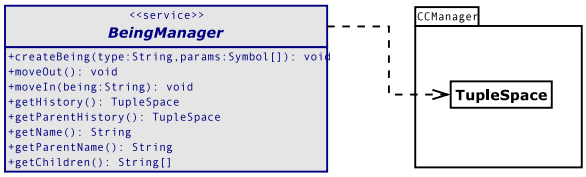


Figura 5.3: Diagrama de classes do serviço *BeingManager*

5.2.2 Suporte às políticas de adaptação

As políticas de adaptação são centrais na estratégia de colaboração entre o *middleware* e a aplicação, quando da execução dos comandos de adaptação por parte do EXEHDA. Estas políticas referem-se às orientações da aplicação para a tomada de decisão do ambiente de execução, o qual controla o comportamento geral da aplicação, buscando deste modo, o equilíbrio entre transparência (uso de políticas gerais) e a participação da aplicação na gerência do processo de adaptação. A adaptação automática contempla aspectos de propósito-geral não considerando as necessidades peculiares de cada aplicação. O usuário quando conhecedor do comportamento da aplicação pode otimizar o processo de adaptação, orientando a estratégia a ser empregada em relação a componentes específicos da aplicação.

Logo, as políticas podem ser de natureza global - válidas para toda aplicação; ou de natureza específica - válidas para alguns componentes em particular. Políticas específicas são semelhantes às globais, diferenciando-se pela identificação do componente (ente) ao qual se referem. Como exemplos de políticas globais especificadas no ambiente de desenvolvimento da linguagem ISAMadapt, têm-se:

- *DISCONNECTION migrate* <*node*> - indica que antes da operação de desconexão ser realizada, os entes da aplicação em execução no equipamento devem ser migrados para outro nodo, o qual é especificado no parâmetro <node>. O parâmetro <node> é uma definição abstrata do nodo, através da qual são estabelecidas características que devem ser atendidas pelo recurso que irá receber os entes a serem migrados. Um exemplo de característica é manutenção do estado de conectividade daquele dispositivo;
- *DISCONNECTION buffering* - indica para chavear as operações de E/S para (cache) *buffer* local antes de desconectar.

Exemplos de política específica são:

- *CLONING ente1 anchorOn being:ente2* - indica que a criação do *ente1* deve ser realizada no nodo onde o *ente2* está localizado. Além disso, caso o *ente2* migre, o *ente1* deve ser migrado juntamente;
- *CLONING ente1 on resource:mobileHost static* - diz que o *ente1* deve ser criado no dispositivo móvel (*default* é criar no nodo local) e que o EXEHDA não pode migrá-lo por decisão própria (parâmetro *static*). Os entes declarados como estáticos normalmente são responsáveis pelo gerenciamento de recursos específicos dos nodos aonde são criados;
- *CLONING ente1 on resource:BD1.rsc* - diz que o *ente1* deve ser criado no nodo que contém o recurso `BD1`. Neste caso, é usado o serviço *Discoverer* do EXEHDA para localizar um EXEHDAnodo que disponibilize o recurso descrito em `BD1.rsc`.

Essas políticas são representadas na forma de um documento XML, o qual é utilizado pelo EXEHDA na parametrização dos serviços que atendem a aplicação. Um exemplo de documento XML para definição de políticas referentes ao escalonamento dos entes é apresentado na figura 5.4.

5.2.3 O suporte ao contexto

Na linguagem ISAMadapt, usa-se a construção *context* para declarar um elemento de contexto a cujo estado o ente da aplicação se adapta. Por exemplo:

```
context  processor::idle.
```

O ente de uma aplicação pode controlar o recebimento de notificações de alteração no estado de um elemento de contexto através dos comandos `suspendNotify` e `resumeNotify`, que suspendem e retornam o recebimento de notificação, respectivamente. A aplicação pode, ainda, consultar o estado de um elemento de contexto através dos comandos `getContext` – retorna o valor contido no serviço, e `checkContext` – força uma monitoração do contexto antes de retornar seu estado. Estes comandos são mapeados para chamadas ao serviço *AdaptEngine* do EXEHDA.

Na fase de desenvolvimento, o programador especifica como deve ser construído o dado referente a cada um dos elementos de contexto utilizados pela aplicação. Estas definições dão origem a um documento XML (`context.xml`), que é interpretado pelo serviço *AdaptEngine* do *middleware*, que por sua vez emprega estas definições na parametrização do serviço de reconhecimento de contexto do EXEHDA (*ContextManager*, vide seção 4.6.3).

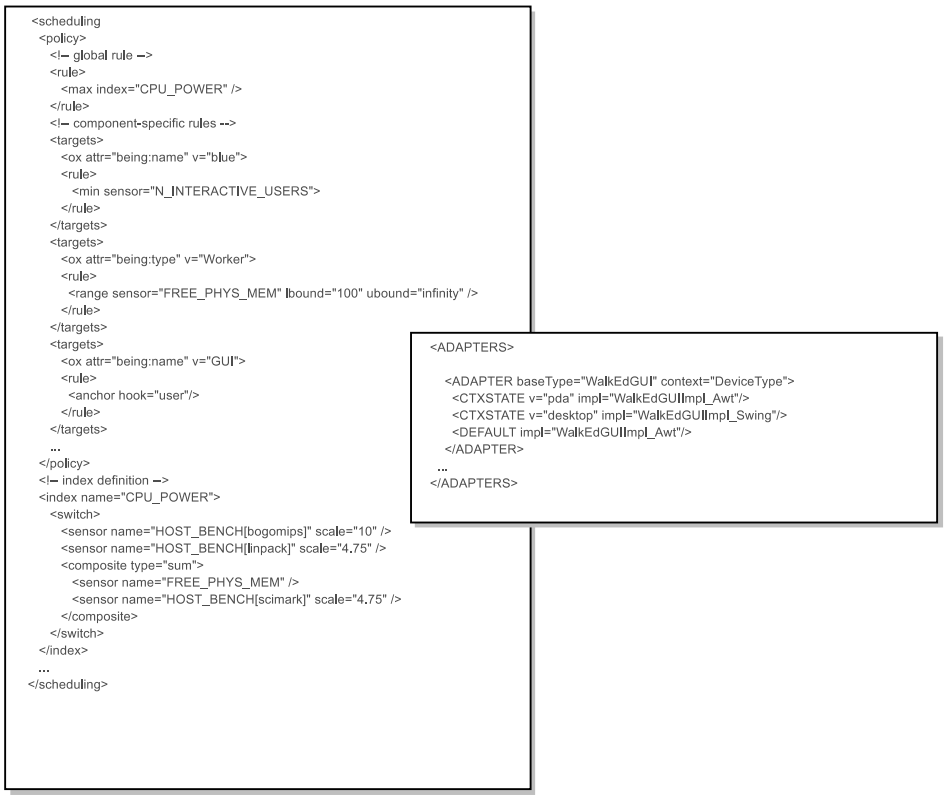


Figura 5.4: Políticas de adaptação não-funcional e funcional

5.2.4 O suporte aos entes e métodos adaptativos

Ente adaptativo é aquele cujo código a ser executado é determinado pelo contexto. Por exemplo, o fragmento de código para um ente `GUI` cuja implementação é dependente do tipo de dispositivo que dispara sua criação é apresentado na figura 5.5.

Neste fragmento de código ISAMadapt é declarado que, após a criação do ente `GUI` (comando `clone`), é carregado o código correspondente ao estado do elemento de contexto `deviceType`.

```
being holo
    {
      holo ( ) {
          clone (GUI);
      }
    }
    // graphic interface adapted to display device
    //
adaptive being GUI
        context deviceType::(laptop, PDA)
{   // shared methods     }
```

Figura 5.5: Código fonte do ente adaptativo

A implementação dos códigos alternativos para o ente é feita através de adaptadores (`adapters`). A figura 5.6 exemplifica a declaração de um adaptador para o ente GUI, que codifica o tratamento da interface gráfica considerando um dispositivo PDA.

A adaptação de código representada pelos entes adaptativos reflete-se especialmente na execução dos comandos `clone` e `move` do ISAMadapt, cujo suporte é detalhado na seção 5.2.5.

Instanciação e migração de entes são processos adaptativos decorrentes da interação do serviço *Executor* com serviços *AdaptEngine* e *ContextManager*. Desta interação resulta a carga contextualizada do código do adaptador. Esta carga é parametrizada pelas alternativas de código definidas na etapa de desenvolvimento, as quais encontram-se registradas na política de adaptação funcional (`adapters.xml`).

```
//@context: deviceType::PDA
// file: GuiPDA.adp
//
adapter being Gui::GuiPDA( )
 {
     // Constructor
     GuiPDA( )  {
         native Java {*
          // use java.awt components to build GUI
          // and register menu callbacks
        *}
      }
.............
 }
```

Figura 5.6: Adaptador do ente GUI para o dispositivo PDA

Os serviços *AdaptEngine* e *ContextManager* são ainda responsáveis pelo controle de adaptações dinâmicas decorrentes de novas alterações no estado dos elementos de contexto. A estratégia de geração de código que permite a ligação entre o modelo de adaptação ISAMadapt e o EXEHDA é exemplificada na figura 5.7.

O ISAMadapt também permite a adaptação no nível de método. Um método adaptativo é um método cuja funcionalidade é adaptativa ao contexto, comporta-se como uma função genérica. Estes métodos são identificados no código pelo modificador `adaptive` acrescido ao nome do método declarado. A implementação do método é composta por códigos alternativos (adaptadores) que especializam determinados comportamentos projetados com base nos diferentes estados possíveis do elemento de contexto de interesse da aplicação.

Por exemplo, a declaração de método adaptativo

```
adaptive outResult (string)  context  deviceType;
```

especifica que o código do método `outResult` será implementado através de adaptadores (`adapter`) para cada estado do contexto `deviceType`. A codificação do adaptador correspondente a cada elemento de contexto é similar à codificação de adaptadores de entes, porém identificando o ente ao qual o método está associado. O controle da adaptação no nível de método transcorre de forma análoga à realizada para entes adaptativos.

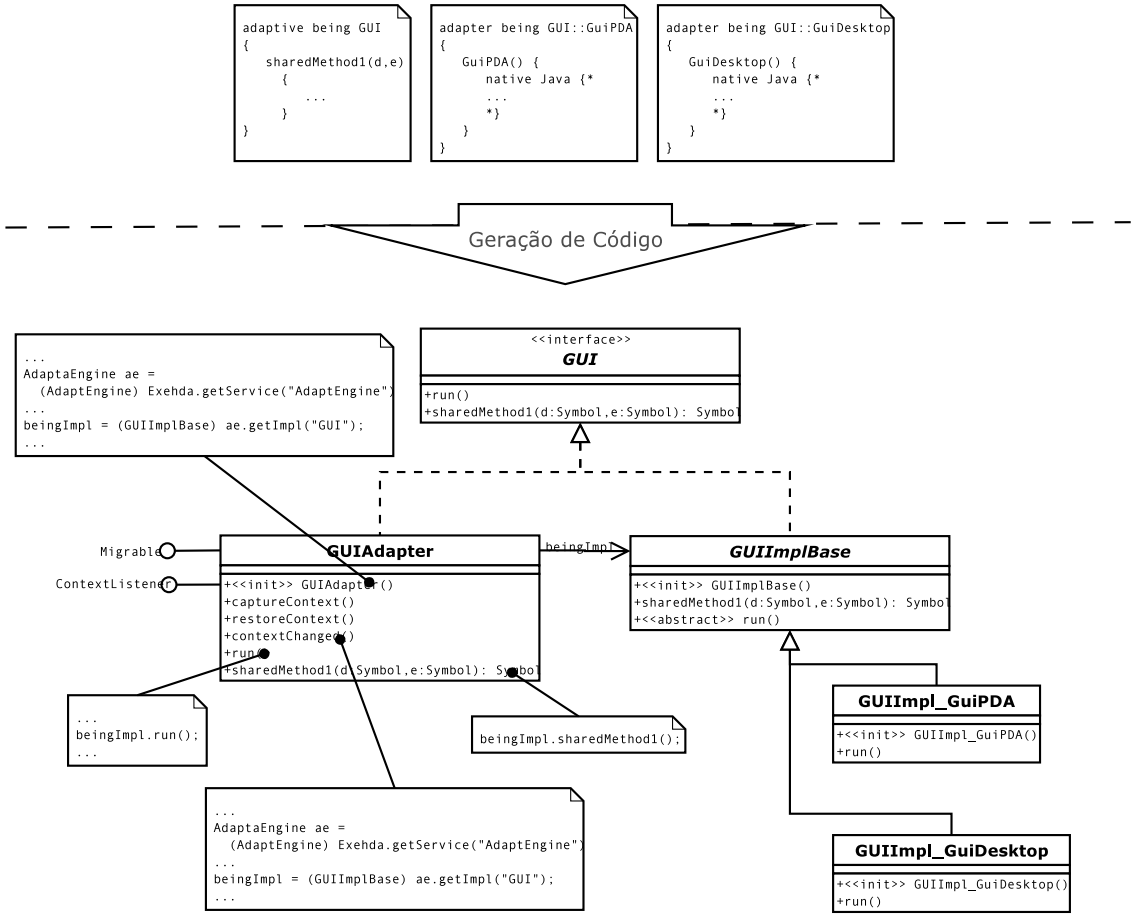


Figura 5.7: Relacionamento EXEHDA e ISAMadapt construído no momento da geração de código

Um aspecto a ser ressaltado é o fato que no EXEHDA o gerenciamento da adaptação não é ativado toda vez que o componente da aplicação (ente ou método) é invocado. Esta característica é decorrente de uma combinação da estratégia de geração de código empregada com o modelo *publish-subscribe* para notificação dos elementos de contexto. Os serviços envolvidos com a gerência da adaptação, como um todo, somente são ativados em função de notificações decorrentes de alterações nos elementos de contexto de interesse da aplicação.

5.2.5 O suporte aos comandos de adaptação

ISAMadapt fornece comandos para modelar estratégias de adaptação mais usuais em aplicações móveis. A implementação dos comandos é gerenciada pelo EXEHDA, com consulta às políticas de adaptação (seção 5.2.1) definidas pelo programador da aplicação. Os comandos fornecidos são: `move`, `clone`, `disconnect`, `reconnect`, `install`, `reschedule`, `discovery`, `onContext`.

clone

O comando `clone` implementa a criação de entes.
Sintaxe[1]:

[1] A sintaxe segue a notação BNF estentida utilizada pelo JAVACC – software para gerar compilador em Java. As notações utilizadas indicam: | → alternativas , ? →opcional, * → zero ou mais ocorrências.

```
clone (cloned_being, clone_being)
      (on|anchoredOn|closerTo) (being:name|resource:name);
```

Por exemplo,

```
clone (worker, #worker_id) anchoredOn  being:master ;
```

indica que o ente `worker` é criado e este fica ancorado pelo ente `master`, ou seja, quando o ente `master` migrar, o ente `worker` também migra, para manter a proximidade física.

Os qualificadores `on`, `anchoredOn` e `closerTo` dão aos mecanismos de gerenciamento físico do EXEHDA a indicações de onde o ente deve ser criado, enquanto que `being:` e `resource:` indicam o elemento lógico que complementa o qualificador. O qualificador `on` designa que o novo ente deve ser criado no mesmo dispositivo do ente ou recurso especificado, `anchoredOn` designa que o ente criado deverá acompanhar o movimento do ente ou recurso especificado, `closerTo` designa que o ente deve ser criado próximo ao ente ou recurso especificado, mas que não acompanha seu movimento.

Deste modo, o nodo onde o ente é fisicamente criado é determinado pelo EXEHDA, a partir das orientações indicadas nas políticas para escalonamento e clonagem, ou expressas diretamente no comando `clone`. Se nada for especificado, a seleção do dispositivo que receberá o ente seguirá a política global especificada para a aplicação.

A interação entre os principais serviços envolvidos no suporte ao comando `clone` está apresentada na figura 5.8. Em resumo, o serviço *Executor* é utilizado para criação remota dos entes. No processo de criação remota o *Executor* interage com o serviço `BDA` para instalação de código sob demanda. O destino da criação remota é decidido em conjunto com o serviço *Scheduler*, considerando as políticas de escalonamento providas pela aplicação. No nodo destino, a implementação do ente (adaptador) é selecionada pelo serviço *AdaptEngine*, considerando para isto a informação de contexto (por exemplo o tipo de dispositivo). O serviço *CCManager* provê o espaço de tuplas distribuído para o ente criado, sobre o qual é mapeada a abstração história.

move

Este comando fornece tanto a mobilidade lógica do ente para dentro de outro, quanto a indicação ao serviço *Scheduler* que o ente irá ser submetido a um procedimento de migração de código. Esta operação é realizada, em geral, para otimização dos aspectos de comunicação inter-entes.

Sintaxe:

```
move (to|closerTo)
     (parent|being: ente_destino |resource:name);
```

por exemplo, `move to parent` indica que o ente é movido logicamente para o ente hierarquicamente superior (seu pai). Neste caso, não é garantido que o ente será alojado em outro equipamento, pois a ocorrência disto é regida por outros parâmetros considerados pelo EXEHDA, em algumas situações, inclusive, são consideradas as condições do contexto. Por exemplo, para forçar uma migração de código para o equipamento onde está o ente pai, deve-se usar o comando `move closerTo parent.`

A implementação do comando `move` na perspectiva do EXEHDA ocorre de forma análoga ao comando `clone` a quando mobilidade física é necessária. Nos casos onde

ocorre apenas a mobilidade lógica, apenas o serviço *BeingManager* e, indiretamente, o serviço *OXManager*, são envolvidos.

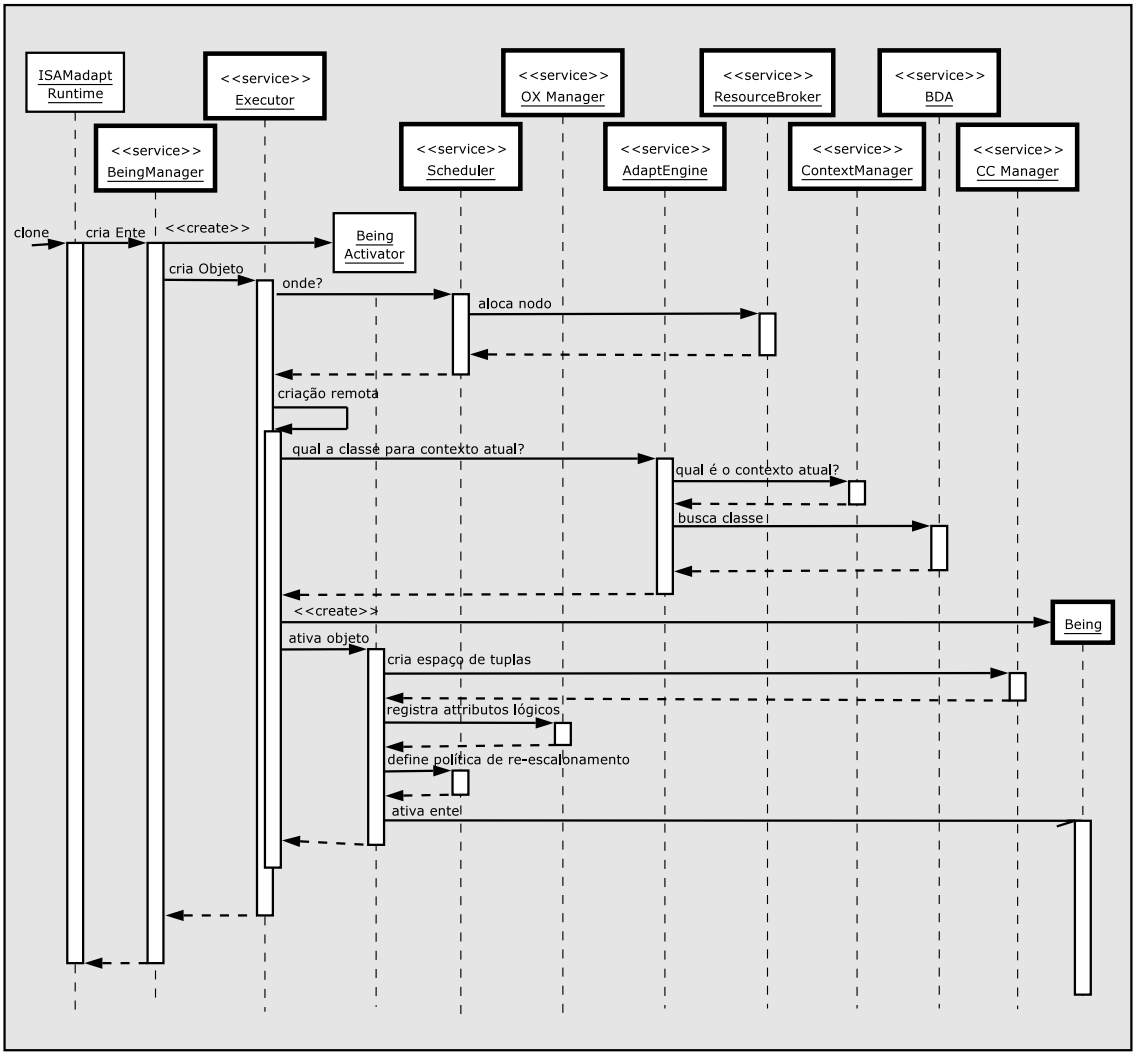


Figura 5.8: Serviços associados a implementação do comando `clone`

reschedule

Permite ao programador influenciar o escalonamento físico, através da modificação das políticas correntes de escalonamento, tanto as globais como as específicas para determinados entes. Este aspecto é importante em aplicações do domínio da Computação em Grade. Através deste comando, sob certas condições de contexto, definidas pelo programador, a aplicação pode forçar um re-escalonamento para melhorar o desempenho.

Sintaxe:

```
reschedule (all|being:name)?(policy:rule)?;
```

onde, `rule` indica a regra que passa a ser utilizada a partir da emissão do comando `reschedule`. As regras já estão pré-definidas na política de escalonamento associada à aplicação (`policies.xml`), sendo ativadas por ocasião deste comando (vide figura 5.3). Os serviços envolvidos estão representados na figura 5.7.

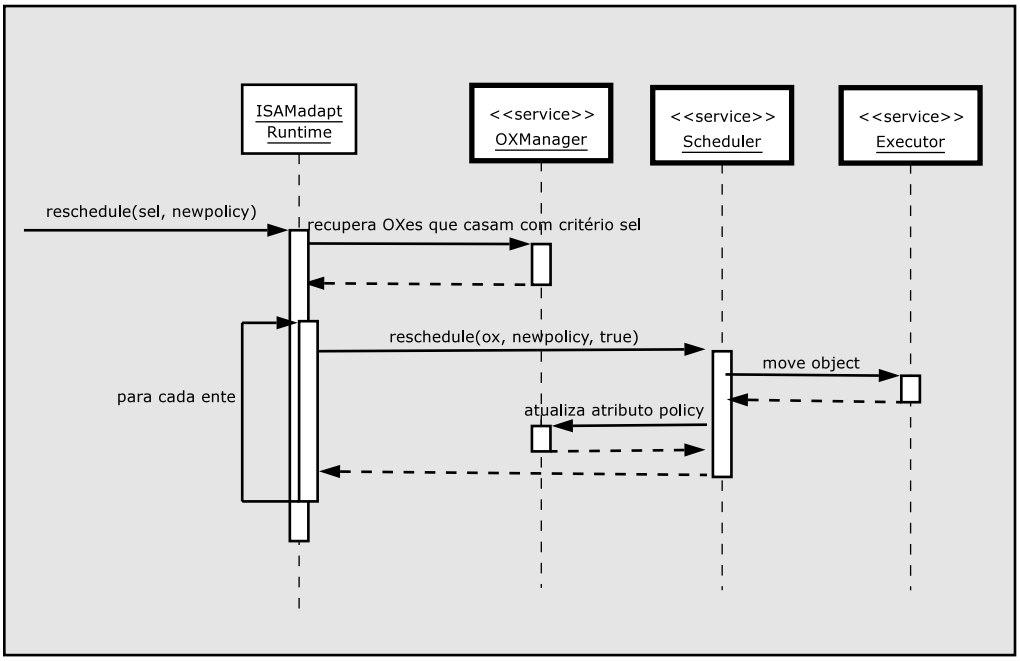


Figura 5.9: Serviços associados a implementação do comando `reschedule`

A implementação do comando `reschedule` é desempenhada pela cooperação entre os serviços *Scheduler* e *OXManager*. A partir do *OXManager* são recuperados os atributos de execução correspondentes a cada um dos entes especificados no comando `reschedule`. A seguir, o serviço *Scheduler* considerando a política selecionada é acionado para reavaliar a concordância ou não da localização corrente destes entes (OXs). Quando necessário, o serviço *Scheduler* aciona o serviço *Executor* para promover a migração do ente para outro nodo.

disconnect

Este comando sinaliza a desconexão lógica da rede, sempre voluntária. A desconexão física da rede é realizada em dois passos: (*i*) a aplicação indica a possibilidade de desconexão, ou seja, realiza uma desconexão lógica; (*ii*) o EXEHDA desconecta fisicamente o dispositivo quando todas as aplicações em execução neste estiverem desconectadas logicamente. A aplicação pode continuar a executar, embora a comunicação com os entes residentes no nodo desconectado fique postergada. O ente mantém o acesso à história local, e quando o nodo for reconectado esta será sincronizada pelo serviço de comunicação e coordenação do EXEHDA (*CCManager*).

Sintaxe:

```
disconnect;
```

O comando disconnect é mapeado para uma chamada ao serviço *AdaptEngine*. Este, por sua vez, conduz um protocolo de duas fases para implementação da desconexão: (*i*) inicialmente o estado do elemento de contexto é definido como “desconexão em curso”, permitindo que os componentes realizem as operações necessárias para operarem no modo desconectado, (*ii*) a seguir, o elemento de contexto é definido como “desconectado”, indicando aos componentes registrados que o processo de conexão foi efetivado. Este procedimento é ilustrado na figura 5.10. Nesta figura, cabe salientar que

a interface `ContextListener` é implementada por vários serviços, inclusive o *CCManager*.

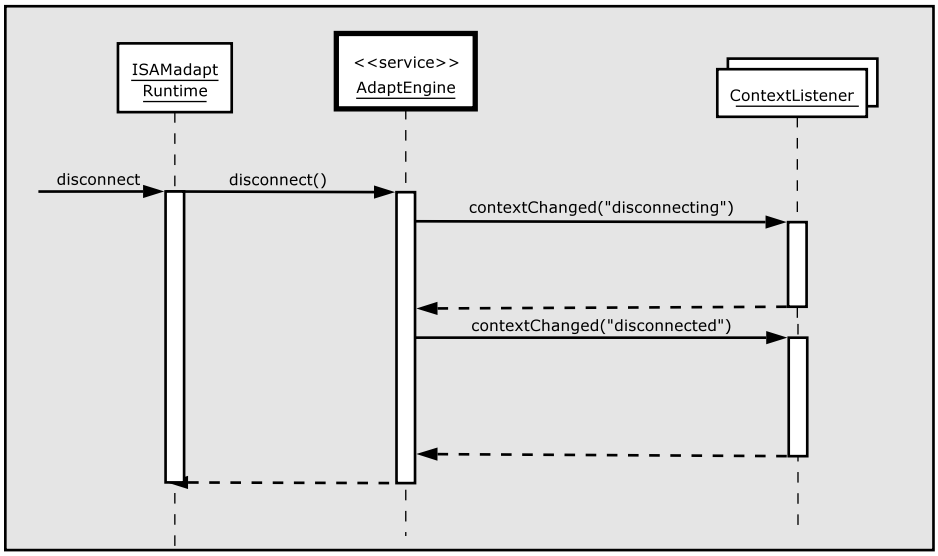


Figura 5.10: Serviços associados a implementação do comando `disconnect`

reconnect

Sintaxe:

```
reconnect;
```

O comando indica a necessidade de conexão física à rede. Este comando é mapeado para uma chamada ao serviço *AdaptEngine*, de forma análoga e complementar ao comando `disconnect`. A reconexão dispara a notificação dos componentes registrados como sendo sensíveis ao elemento de contexto "estado de conexão".

install

Permite disponibilizar uma aplicação no AVU (Ambiente Virtual do Usuário). Este procedimento não implica em cópia do código executável da aplicação; quando da execução, o código da aplicação será buscado pelo serviço `BDA`.

Sintaxe:

```
install <applicationName>
      at <idUser> (, <idUdser>)* ([start])?  ;
```

onde, `idUser` define a identificação do usuário cujo ambiente virtual conterá a nova aplicação. Se o qualificador `start` for declarado, a aplicação é executada automaticamente se o nodo está conectado; caso contrário, esta será executada quando o nodo se conectar.

Na implementação desta semântica, o serviço *SessionManager* é acionado para modificar a sessão padrão do usuário destino, incluindo nesta uma aplicação do sistema que concluirá a instalação da aplicação, mediante o consentimento do usuário.

discovery

Permite solicitar ao *middleware* uma descoberta de recurso. O resultado é atribuído à variável definida no comando. Esta variável é uma lista que contém a referência aos serviços/recursos encontrados, e poderá ser armazenada na história do ente para uso posterior.

Por exemplo:

```
discovery BD (“avu:BD1.rsc”)
      //define o nome do arquivo de definição do recurso
      {
         history!BD; // grava o resultado na história do ente
         while (history#?host)  {
         // enquanto leitura não bloqueante
         //retornar referência ao nodo
           clone (worker, #worker_id) anchorOn  resource:host;
        // cria o ente no nodo que contém o recurso, migra o
       // ente se o recurso migrar
        }
      }
```

No exemplo, o comando `discovery` é emitido para a procura de Bases de Dados com os atributos definidos no arquivo `BD1.rsc`, armazenado no Ambiente Virtual do Usuário. Esta definição alimenta o serviço *Discoverer* do EXEHDA, o qual procede a localização do recurso.

onContext

O comando `onContext` agenda um bloco de comandos que serão executados quando um determinado contexto tornar-se disponível. Este comando apresenta uma semântica assíncrona que permite ao ente continuar executando suas tarefas enquanto comandos podem ficar à espera do contexto que necessitam. A implementação deste comando está associada ao registro de ações junto ao serviço *AdaptEngine* do *middleware*.

O próximo trecho de código exemplifica a criação do ente worker quando um processador se tornar disponível. Neste exemplo, o comando clone ficará bloqueado até receber a notificação de processador disponível, porém o ente continuará sua execução no comando seguinte ao bloco `onContext`. A implementação do comando `onContext` tem por base a combinação de uma estratégia de geração na qual os blocos `oncontext` são mapeados para objetos `Action`. Os objetos `Action` são agendados para execução junto ao serviço *AdaptEngine*, sendo disparados quando o contexto de interesse torna-se disponível.

```
context   processor::idle;
......
onContext processor::idle {
      clone (worker,#worker_id);
}
```

O qualificador `sync`, colocado antes do comando `onContext`, indica uma semântica síncrona, ou seja, o ente ficará bloqueado a espera da informação do contexto.

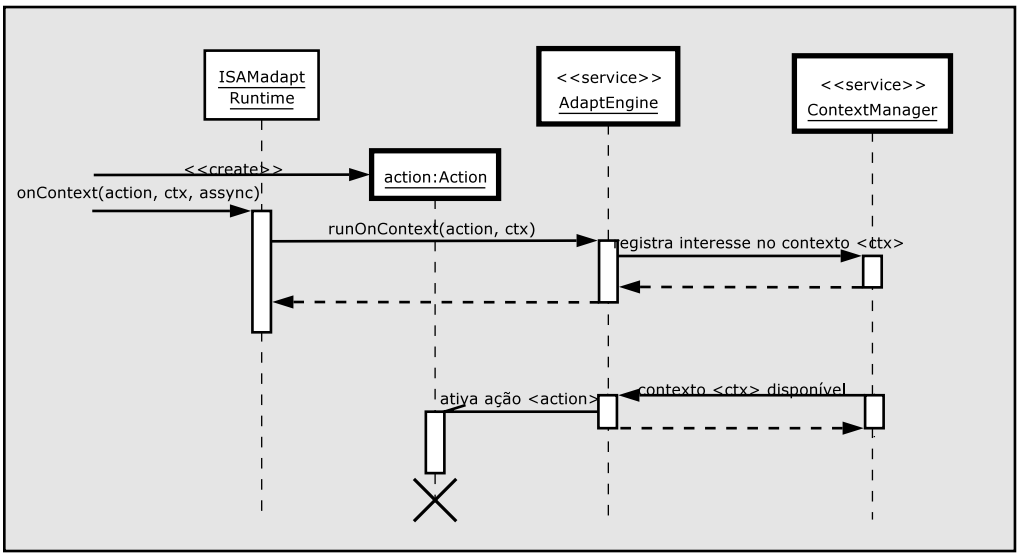


Figura 5.11: Serviços associados a implementação do comando `onContext`

O próximo capítulo concretiza sob a visão do usuário e/ou administrador do ISAMpe como os requisitos e serviços, são instanciados no EXEHDA na execução de aplicações da Computação Pervasiva.

6 DINÂMICA DE GERENCIAMENTO DO ISAMPE E DE EXECUÇÃO DE APLICAÇÕES NO EXEHDA

The aim of science is not to open the door to infinite wisdom,
but to set a limit to infinite error.

- Bertolt Brecht

Este capítulo sintetiza os principais aspectos operacionais do EXEHDA. Esta síntese é feita sob duas óticas nas quais o EXEHDA pode ser visto: (*i*) gerenciador do ISAMpe; (*ii*) provedor de serviços para execução de aplicações adaptativas com mobilidade lógica e física em escopo global, no contexto da Computação Pervasiva.

6.1 Gerenciando o ISAMpe

A organização do ISAMpe e a descrição dos seus principais componentes foram introduzidas na seção 4.1. O EXEHDA assume o papel de gerência da arquitetura distribuída que forma o ISAMpe, a qual é mantida de forma incremental e dinâmica. O ISAMpe, em função dos interesses das instituições envolvidas, do contexto global no nível multi-institucional, bem como do interesse das aplicações, é alterado através da inclusão de novos dispositivos e/ou serviços, ou da exclusão de outros já existentes.

6.1.1 Administradores de EXEHDAcel e de recurso privado

Considerando que o ambiente *pervasivo*, na arquitetura ISAM, é um ambiente *wide-area* (*i*), e sendo neste tipo de ambiente usual a operação ocorrer em uma perspectiva distribuída/multi-institucional, e (*ii*) a decisão de concepção para o EXEHDA de resguardar a autonomia das instituições participantes na definição dos recursos que serão agregados ao ambiente, dois atores se destacam na gerência do ISAMpe:

1. administrador de EXEHDAcel (célula de execução);
2. administrador de recurso privado.

O administrador de EXEHDAcel

O administrador de EXEHDAcel é responsável, no âmbito da instituição, pelas seguintes atividades:

- manter os recursos de uso compartilhado que irão compor a célula de execução, definindo suas respectivas políticas de acesso;
- manter a EXEHDAbase;
- adicionar e remover usuários da EXEHDAcel.

Com o propósito de facilitar as atividades do administrador, foi concebida e prototipada uma ferramenta, direcionada às tarefas de manutenção das células de execução e seus diversos serviços. Esta ferramenta é denominada EXEHDA-AMI (EXEHDA *Architecture Managenment Interface*). Uma visão geral da mesma está apresentada na figura 6.1.

A EXEHDA-AMI segue a filosofia incremental presente na composição do *middleware* como um todo. Neste sentido, ela é constituída por um módulo-base, ao qual podem ser agregados dinamicamente componentes que ampliam suas funcionalidades, sem necessidade de recompilação do núcleo em uso. Os principais componentes são apresentados ao longo deste capítulo.

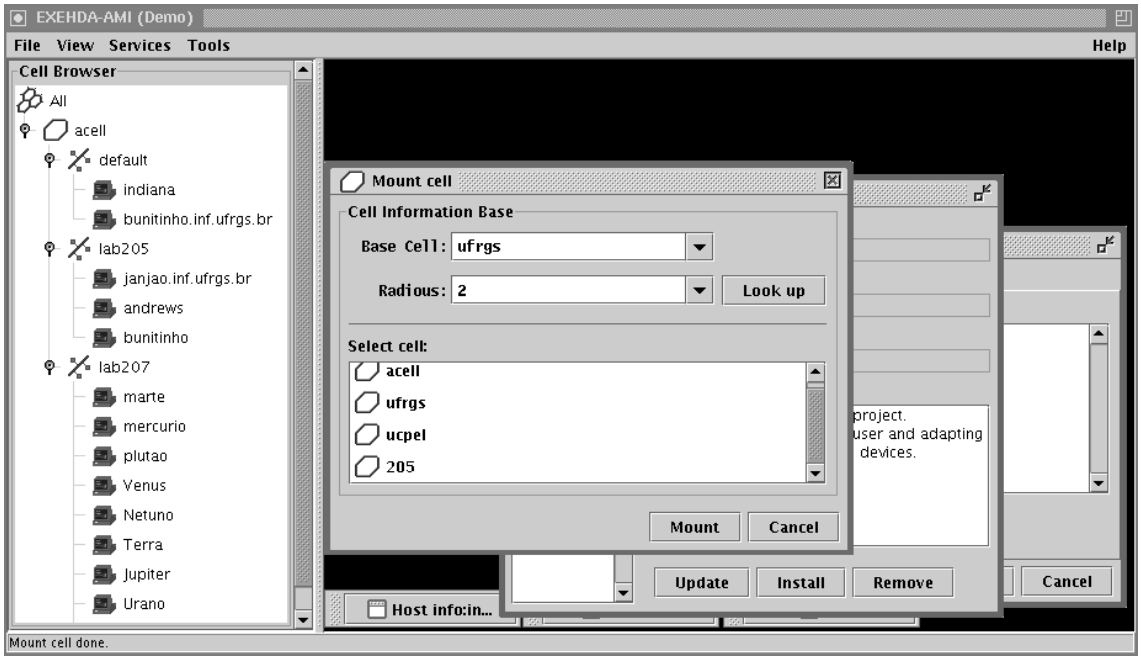


Figura 6.1: Visão geral da ferramenta EXEHDA-AMI

Uma das funcionalidades oferecidas pela EXEHDA-AMI é o suporte à navegação pelas células atualmente ativas no ISAMpe, permitindo uma inspeção das características dos EXEHDAnodos que compõem as EXEHDAcels. Esta funcionalidade é provida pelo módulo *Cell Browser* ilustrado nas figura 6.1 e figura 6.2.

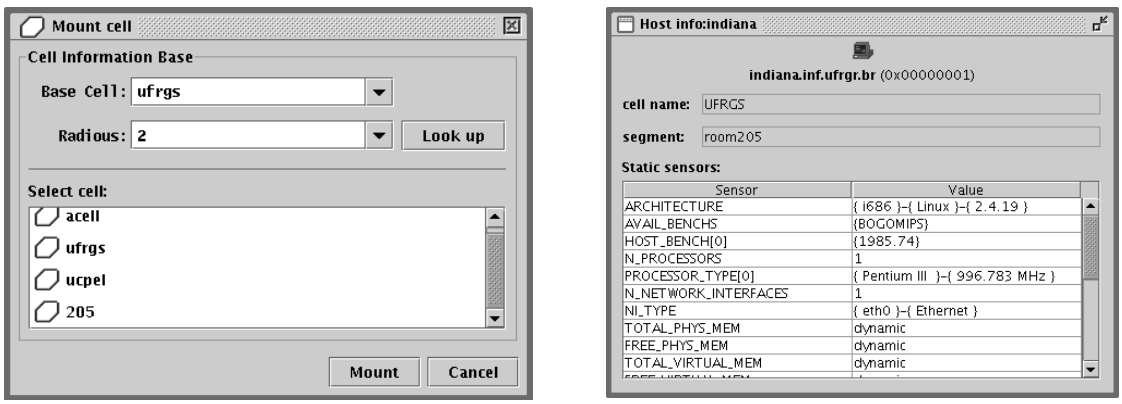


Figura 6.2: EXEHDA-AMI: suporte à navegação nos recursos das EXEHDAcels

O serviço de adição e remoção de usuários de uma EXEHDAcel é provido por um módulo específico da EXEHDA-AMI para esta finalidade (vide figura 6.3). Usuários qualificados como desenvolvedores estão habilitados a instalar e remover aplicações no serviço BDA da EXEHDAcel (vide *checkbox developer*).

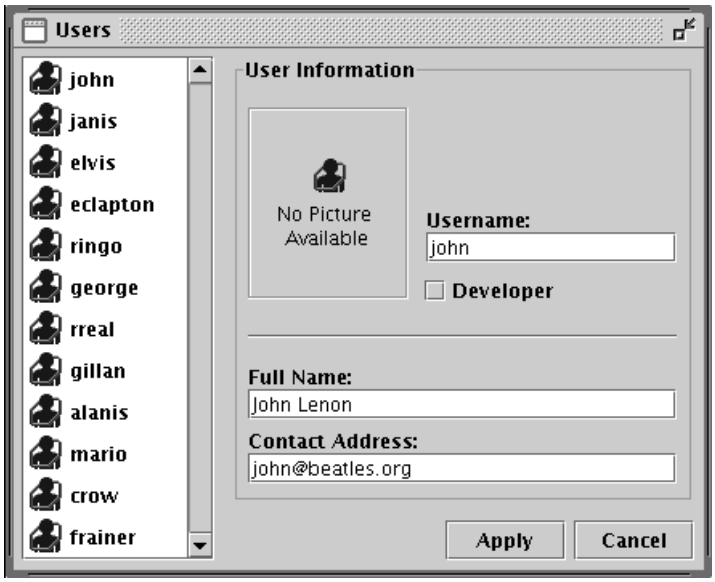


Figura 6.3: Módulo de gerência de usuários da EXEHDA-AMI

O administrador de recursos privados

Por sua vez, o administrador de recursos privados é responsável por manter recursos pertencentes a um usuário (ou grupo de usuários) específico. Suas atribuições se sobrepõem àquelas do administrador da célula, ficando responsável pela manutenção da instalação do *middleware* no escopo de um recurso em particular, bem como pela definição da política de acesso a este recurso. Os possíveis usuários de um recurso privado deverão ser usuários previamente cadastrados na célula. O administrador de um recurso privado não tem autonomia para criar usuários.

6.1.2 Manutenção da EXEHDAbase

Os serviços do *middleware* em execução nos EXEHDAnodos cooperam e executam de forma coordenada com aqueles em execução na base, para a construção de semânticas distribuídas. Como apresentado na seção 4.3, existe, para os serviços, uma visão de instância de nodo e de instância celular. Esta abordagem é fundamental para que, ao mesmo tempo em que se mantém a consistência da operação distribuída, os serviços do *middleware* permaneçam operacionais nos nodos mesmo em períodos de desconexão.

A manutenção da EXEHDAbase de uma célula ISAMpe consiste na configuração dos atributos gerais da célula, e na gerência das instâncias celulares dos serviços do *middleware*. Os procedimentos envolvidos na instalação, configuração e inicialização da EXEHDAbase são:

- **instalação**: realizada com a cópia dos arquivos correspondentes ao código executável dos serviços celulares necessários à operação da

EXEHDAbase. Atualmente, o software necessário para instalação da EXEHDAbase é disponibilizado na forma de um CD-ROM;

- **configuração:** corresponde à construção do perfil de execução do *middleware* nos equipamentos que constituem a EXEHDAbase, com a respectiva parametrização dos serviços que vão ser disponibilizados para a célula de execução. Tipicamente, o perfil de uma EXEHDAbase inclui os parâmetros para ativação das instâncias celulares dos serviços do *middleware*;
- **inicialização:** corresponde ao disparo da execução do *middleware* nos equipamentos, indicando o perfil de serviços desejado.

No que diz respeito à configuração dos atributos da EXEHDAcel, este procedimento inclui a configuração das vizinhanças estática e dinâmica da célula, de forma a integrá-la às demais células que formam o ISAMpe. Para este procedimento, existe um módulo específico da EXEHDA-AMI (vide figura 6.4).

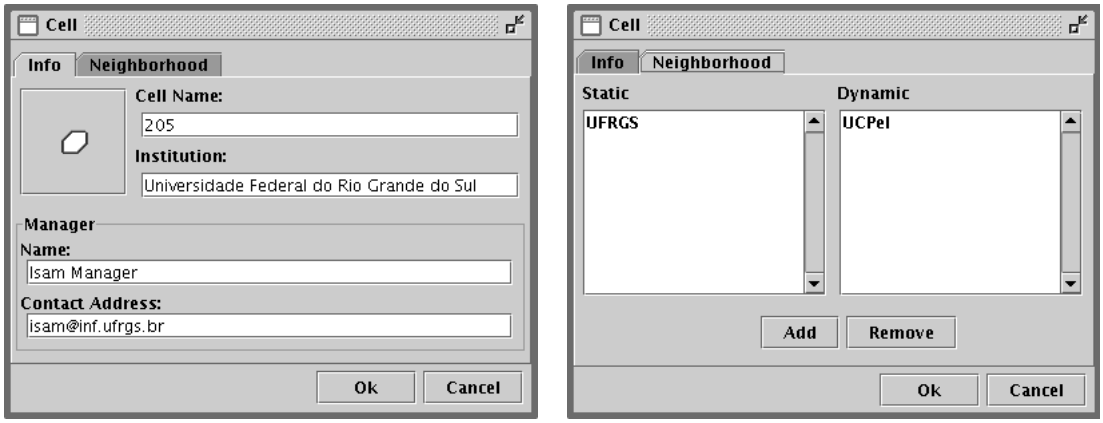


Figura 6.4: Módulo de configuração de atributos da EXEHDAcel

Com o intuito de atender à elevada dinamicidade com que podem ser agregados/desagregados nodos no ISAMpe (grade pervasiva), foi concebido um serviço específico no EXEHDA para atender esta característica. Este serviço é o DC (*Dynamic Configurator*), o qual permite a configuração de forma automática dos EXEHDAnodos, conforme apresentado na seção 4.5.9. A configuração da instância celular deste serviço em execução na EXEHDAbase é feita pelo módulo apresentado na figura 6.5.

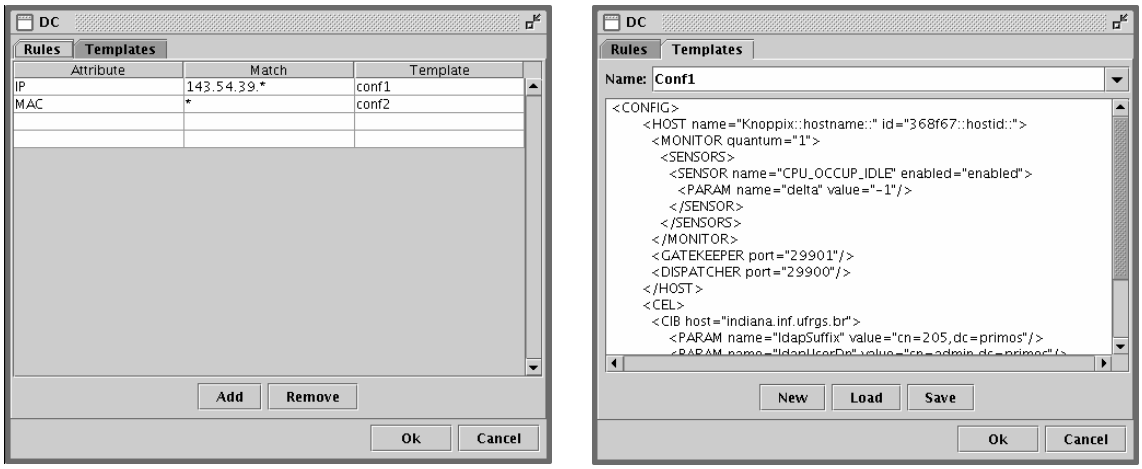


Figura 6.5: Módulo de configuração do serviço DC

A instalação de software é uma prerrogativa do usuário desenvolvedor, porém, em algumas situações, pode ser oportuno ao administrador da EXEHDAcel instalar ou remover aplicações no serviço BDA. O módulo que permite a gerência do conteúdo da BDA pelo administrador é apresentado na figura 6.6.

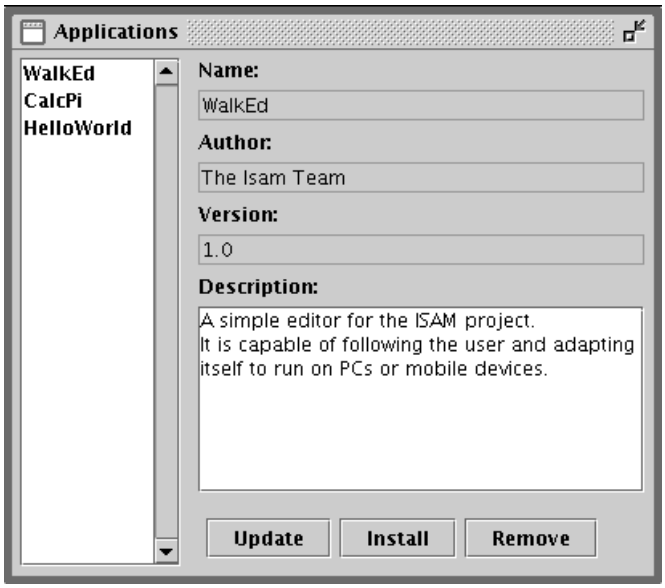


Figura 6.6: Módulo do administrador de EXEHDAcel para gerência de aplicações da BDA

Observe que, como abordado na seção 4.1, a EXEHDAbase, apesar de logicamente ser uma entidade, fisicamente pode ter seus serviços distribuídos em vários equipamentos por aspectos de escalabilidade.

6.1.3 Incorporando recursos à EXEHDAcel

Os recursos que compõem uma célula podem ser de (*a*) processamento, utilizados para execução das aplicações em geral ou (*b*) especializados, que não executam diretamente as aplicações, mas provêem a estas algum tipo de funcionalidade. Sua

natureza pode ser hardware e/ou software: impressão, armazenamento persistente, bancos de dados, etc.

Em uma visão homogênea, os recursos seriam gerenciados pelo *middleware* sob todos os aspectos. Isto, entre outras conseqüências, facultaria uma melhor gerência das políticas de acesso [FOS 2001]. A perspectiva de o *middleware* ser a única forma de intermediar acesso aos recursos, no entanto, comprometeria a visão da agregação incremental de novos modelos de equipamentos, pois exigiria previamente, para cada um destes, a concepção e a implementação do mecanismo de gerência necessário.

A opção feita quando da concepção do EXEHDA foi o acesso a recursos especializados, via de regra, não ser intermediado pelo *middleware*. Tais recursos, no entanto, são catalogados junto à base de informações da célula (vide serviço CIB, seção 4.5.2) de forma que possam ser localizados e acessados pelas aplicações em execução. Na figura 6.7 pode ser visto o módulo responsável pela tarefa de catalogação.

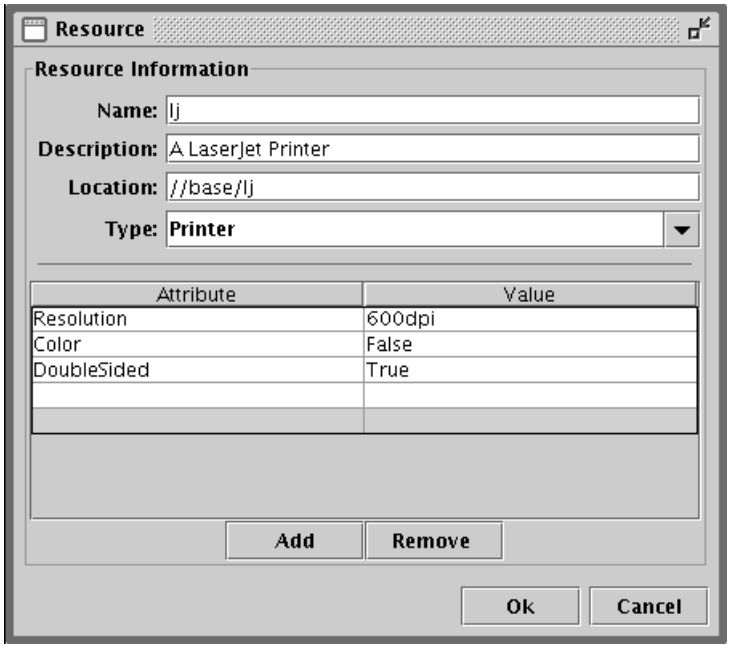


Figura 6.7: Módulo de catalogação de recursos do EXEHDA-AMI

Com relação aos nodos de Processamento, (EXEHDAnodos) a incorporação dos mesmos a EXEHDAcel ocorre de forma similar à EXEHDAbase, conforme as etapas a seguir:

1. **instalação**: consiste na cópia dos arquivos correspondentes ao código executável dos serviços necessários à operação do EXEHDAnodo. Atualmente, o software necessário para instalação do EXEHDAnodo é disponibilizado na forma de um CD-ROM, ou pode ser obtido via *download* disponibilizado no site do EXEHDA [EXE 2003];
2. **configuração**: pode ser feita de forma manual, onde o administrador define quais serviços e sob que condições estes serão disparados para um nodo de processamento em particular; ou de forma automática, pela utilização do serviço DC. O serviço DC está disponível na EXEHDAbase, sendo localizado por um mecanismo de *broadcast*

(vide seção 4.5.9). Como resultado do procedimento de configuração, é gerado um perfil de execução para o *middleware* naquele recurso (vide seção 4.4.1);

3. **inicialização**: consiste no disparo de forma manual ou automática do *middleware* segundo um perfil de execução para o recurso em questão. A forma de implementação do disparo automático é dependente do sistema operacional hospedeiro.

6.2 Gerenciando a execução de aplicações no EXEHDA

Uma característica presente na Computação Pervasiva, é o deslocamento do usuário, com ou sem seu dispositivo portátil, mantendo acesso ao seu ambiente computacional. Nesta seção são apresentados os mecanismos do EXEHDA que facultam essa *semântica siga-me*, especificamente no que diz respeito ao acesso a dados, aplicações, e ao seu respectivo uso. Os mecanismos descritos nesta seção permitem que o usuário dispare ou continue executando suas aplicações independentemente da EXEHDAcel em que esteja fisicamente localizado.

6.2.1 Controle de sessão de usuário no EXEHDA

Estando o usuário com uma sessão de trabalho ativa no EXEHDA, o mesmo estará apto a executar as aplicações de seu interesse. Os comandos de *controle de sessão*, disponibilizados pelo EXEHDA ao usuário ISAM, são necessários para viabilizar a implementação da semântica *siga-me* das aplicações. Os principais, dentre estes comandos, têm suas funcionalidades resumidas a seguir.

Login/Logout

O comando de `login` é responsável pela autenticação do usuário junto ao serviço *Gatekeeper* do nodo, procedimento este que é baseado em um mecanismo de chave pública/privada [MEN 97]. A chave privada é mantida pelo usuário em um meio de armazenamento de sua responsabilidade como, por exemplo, um smartcard. A chave pública, por sua vez, é disponibilizada através de um certificado armazenado no serviço *CIB* da célula onde o usuário foi cadastrado. No caso de `login` em outra célula, que não aquela em que originalmente o usuário foi cadastrado, o serviço *CIB* fará a busca de forma transparente do certificado do usuário. Como resultado de uma operação de `login` realizada com sucesso, a sessão padrão do usuário é ativada. Caso esta sessão inclua aplicações anteriormente interrompidas, estas serão recolocadas em operação. Tipicamente, a sessão padrão do usuário inclui acesso a aplicação ISAM Desktop (seção 6.2.3), a qual permite acesso às demais aplicações instaladas no ambiente virtual daquele usuário.

De forma complementar ao `login`, a operação de `logout` libera os recursos empregados na gerência dos contextos de execução das aplicações que integram a sessão padrão do usuário. Esta liberação implica em informar os serviços de Gerenciamento de Contexto (*ContextManager*) e Máquina de Adaptação (*AdaptEngine*) do término das aplicações, de forma que estes possam liberar os recursos a eles alocados. O estado da sessão padrão é armazenado no Ambiente Virtual do Usuário (AVU), para que possa ser recuperado por ocasião do próximo `login`.

Save/Restore session

Um usuário pode possuir um número arbitrário de sessões, adicionalmente a sua sessão padrão, as quais são gerenciadas por meio dos comandos `save session` e `restore session`. O comando `save session` permite mover as aplicações contidas na sessão padrão para uma sessão alternativa, armazenada no AVU. Mover uma aplicação para uma sessão alternativa é um procedimento análogo ao de mover os componentes da aplicação entre processadores. A principal diferença é que o destino, neste caso, não é um EXEHDAnodo, mas sim um serviço de armazenamento persistente, que manterá o estado dos componentes até que estes sejam recuperados através de um comando `restore session`. O comando `restore session`, por sua vez, desempenha função oposta, permitindo que as aplicações salvas em uma determinada sessão alternativa sejam reincorporadas à sessão padrão do usuário e retomem à execução. A operação dos comandos Save/Restore session está relacionada ao serviço *SessionManager*.

Disconnect/reconnect

Estes comandos, que correspondem a chamadas ao serviço *AdaptEngine*, atuam sobre o estado de conectividade do dispositivo sob controle do usuário, e disparam os procedimentos associados à desconexão planejada. As modificações no estado de conectividade, em resultado da execução de um destes comandos, são percebidas pelo serviço de reconhecimento de contexto (*ContextManager*), tanto em sua instância local ao EXEHDAnodo, quanto em sua instância celular (EXEHDAbase). Desta forma, todos os elementos registrados como sensíveis a este tipo de contexto são notificados, permitindo sua adaptação ao novo estado de conectividade.

A interação do usuário com os comandos de controle de sessão pode ser feita por linha de comando, disparando aplicações específicas para comando de controle; porém, a forma preferencial de manipulação destes comandos é através da aplicação gráfica ISAM Desktop (vide seção 6.2.3).

6.2.2 Disparo de aplicações

Para o EXEHDA toda aplicação é caracterizada pelo seu respectivo *descritor de disparo*. Este descritor é um documento XML gerado durante a fase de desenvolvimento da aplicação. Usualmente seu nome é <*aplication-name*.isam>. O descritor de disparo agrupa uma série de metadados que habilitam a execução da aplicação, a partir de EXEHDAnodos, em qualquer uma das células de execução que integram o ISAMpe. A figura 6.8 exemplifica o descritor de disparo para uma aplicação ISAM.

Entre os metadados incluídos no descritor de disparo da aplicação estão: (*a*) uma descrição da funcionalidade implementada, (*b*) o desenvolvedor, (*c*) os parâmetros fixos utilizados pela aplicação, (d) uma referência, independente de localização (<jar href="bda:/SomeApp.jar"/>), para o arquivo JAR que contém as classes da aplicação. Esta referência é, em tempo de execução, resolvida pelo serviço BDA do EXEHDA, viabilizando a instalação do código executável nos nodos utilizados pela aplicação.

```
<?xml version="1.0" encoding="UTF-8"?>
<isamapp spec="1.0" href="someapp.isam">
    <info>
       <title>ISAM demo application</title>
       <vendor>ISAM team</vendor>
     <description>
         This is a sample XML descriptor for an ISAM demo application
     </description>
     <icon href="someapp.png" />
    </info>
    <code>
        <main class="isam.demo.someapp.Main" />
     <jar href="SomeApp.jar"/>
    </code>
</isamapp>
```

Figura 6.8: Descritor de disparo de aplicação

Não é exigência do EXEHDA que o conteúdo da BDA existente em uma determinada EXEHDAbase seja replicado em todas as células do ISAMpe. Deste modo, o arquivo JAR que contém o código executável de uma aplicação pode estar instalado apenas em uma célula de execução em particular.

Considerando que o EXEHDAnodo no qual a aplicação está sendo disparada pode pertencer a uma célula distinta daquela cuja BDA originalmente disponibiliza o código da aplicação, um acesso intercelular pode ser necessário. Este acesso, porém, é tratado de forma transparente para a aplicação pelo serviço BDA do *middleware*. Este comportamento operacional viabiliza, no que diz respeito ao código sendo acessado, a *semântica siga-me* proposta (vide seção 4.2).

O disparo da aplicação é realizado no âmbito do EXEHDA por um utilitário específico para esta funcionalidade, denominado `isam-run`. Por ocasião de sua ativação, o isam-run interage com o serviço *Gatekeeper*, tipicamente já em execução no mesmo EXEHDAnodo, repassando a este serviço o descritor de disparo da aplicação, estendido pela inclusão dos parâmetros de execução. A figura 6.9 caracteriza o disparo de aplicação através do `isam-run`.

```
cw@guaica$>isam-run isamDesktop.isam A B C D
```

Figura 6.9: Disparo de aplicação via utilitário isam-run

O descritor de disparo ampliado da aplicação, apresentado na figura 6.10, é repassado para o serviço *Gatekeeper*, o qual delega ao serviço *Executor* do nodo o disparo efetivo da aplicação, após validar a requisição.

O serviço *Executor* cria uma instância do *Class Loader* para a aplicação em questão. Esta instância é responsável pela integração do mecanismo de carga dinâmica de código da plataforma Java, ao repositório de código *pervasivo* provido pelo serviço BDA do *middleware*. Na continuidade, o *Executor* inicia o processamento da aplicação pela classe indicada no elemento `<main>` do descritor de disparo.

```
<?xml version="1.0" encoding="UTF-8"?>
<isamapp spec="1.0" href="someapp.isam">
    <info>
        <title>ISAM demo application</title>
     <vendor>ISAM team</vendor>
     <description>
         This is a sample XML descriptor for an ISAM demo application
     </description>
     <icon href="someapp.png" />
    </info>
    <code>
        <main class="isam.demo.someapp.Main" />
     <jar href="bda:/SomeApp.jar"/>
    </code>
    <parameters>
     <param>A</param>
     <param>B</param>
     <param>C</param>
     <param>D</param>
    </parameters>
</isamapp>
```

Figura 6.10: Descritor de disparo de aplicação ampliado

Toda aplicação, quando iniciada, recebe um identificador único (*ApplicationId*) no escopo do ISAMpe, e tem seu descritor de disparo ampliado registrado junto ao serviço CIB. Desta forma, é viabilizada a recuperação dos atributos de execução para novos EXEHDAnodos que venham a ser incorporados à execução distribuída daquela aplicação.

6.2.3 A aplicação ISAM Desktop

O disparo manual de aplicações, descrito no item 6.2.2, é um mecanismo básico através do qual podem ser ativadas aplicações pervasivas. O `isam-run` não é uma aplicação pervasiva e conseqüentemente não goza de características definidas para ela. Dentre outros aspectos, o isam-run não pode ser instalado sob demanda, nem ser objeto da semântica *siga-me*. Sua disponibilização é feita junto com o núcleo mínimo do *middleware* que deve ser instalado no EXEHDAnodo. A priori o `isam-run` deve ser utilizado para disparar uma aplicação pervasiva que habilite o usuário a manipular seu AVU (vide seção 4.8.2).

Tipicamente, a primeira aplicação efetivamente pervasiva com a qual o usuário interage é o utilitário ISAM Desktop, cuja interface gráfica é adaptada ao estado do elemento de contexto: “tipo de dispositivo em uso”. Por exemplo, na figura 6.11 é apresentada uma visão do ISAM Desktop elaborada para o PDA Sharp Zaurus 5600. A configuração mostrada diz respeito ao usuário “eTester”, utilizado para testes do comportamento do EXEHDA.

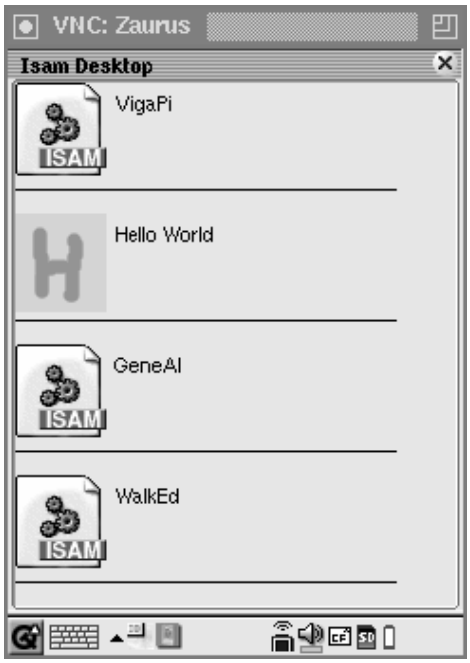


Figura 6.11: ISAM Desktop no PDA Zaurus

O ISAM Desktop provê acesso às aplicações instaladas pelo usuário em seu ambiente virtual. Observe que o conceito de aplicação instalada no ambiente virtual não implica que o código da aplicação seja inserido no AVU do usuário. De fato, apenas o descritor de disparo de aplicação é inserido no AVU, sendo o código da aplicação recuperado sob demanda a partir da BDA no momento em que a aplicação começa a ser executada. O conjunto de aplicações instaladas no AVU forma o **"**Desktop Pervasivo do Usuário". Fisicamente, este "Desktop Pervasivo" corresponde a um arquivo XML armazenado sob um nome `contents.xml` dentro de AVU do usuário - *avu:/Desktop/contents.xml*.

O conteúdo do arquivo `contents.xml` é formado pela tag <info> extraída dos descritores das aplicações instaladas, como mostrado na figura 6.12. Adicionalmente, para cada aplicação instalada, é incluída uma referência para o descritor de disparo daquela aplicação. Com isto fica habilitada a recuperação das informações necessárias quando do disparo da aplicação (por exemplo, a referência para BDA que contém o código da aplicação). O formato do `contents.xml` está ilustrado na figura 6.12.

```
<desktop>
   <isamapp href="avu:/Desktop/vigapi.isam">
      <info>
        <title>VigaPi</title>
        <vendor>ISAM team</vendor>
        <description>
           Civil Engineering Application
        </description>
        <icon href="avu:/Desktop/ISAM_app_default.png">
      <info>
   </isamapp>
   <isamapp href="avu:/Desktop/helloworld.isam">
      <info>
        <title>ISAM HelloWorld demo application</title>
        <vendor>ISAM team</vendor>
        <description>
           Shows Hello world message in the display.
        </description>
        <icon href="avu:/Desktop/helloworld.png">
      <info>
   </isamapp>
   <isamapp href="avu:/Desktop/geneal.isam">
      <info>
        <title>Gene Alighment</title>
        <vendor>ISAM team</vendor>
        <description>
           Search for similar Gene sequences.
        </description>
        <icon href="avu:/Desktop/ISAM_app_default.png">
      <info>
   </isamapp>
   <isamapp href="avu:/Desktop/walked.isam">
      <info>
        <title>WalkEd</title>
        <vendor>ISAM team</vendor>
        <description>
           Pervasive Walking Editor.
        </description>
        <icon href="avu:/Desktop/ISAM_app_default.png">
      <info>
    </isamapp>
 </desktop>
```

Figura 6.12: Descritor do ISAMdesktop

6.3 Uso do EXEHDA a partir de um Live-CD

O software da plataforma EXEHDA também é disponibilizado na forma de um live-CD (vide figura 6.13). Esta iniciativa tem por objetivo permitir que novos usuários experimentem de forma confortável e rápida a plataforma ISAM, sem necessidade de instalar no seu equipamento toda a infra-estrutura de software necessária. É importante observar que neste CD os serviços EXEHDA estão previamente configurados e personalizados para o sistema operacional hospedeiro utilizado.

Figura 6.13: ISAM live-CD

Nesta tecnologia, tanto o sistema operacional quanto o software correspondente ao EXEHDA são alojados em um CD, a partir do qual o equipamento que irá fazer parte do ISAMpe realizará sua inicialização (*boot*).

O live-CD do ISAM foi projetado tendo por base o Knoppix [KNO 2003]. A seleção do Knoppix Linux foi decorrência de: (*i*) sua ampla aceitação internacional; (*ii*) ser baseado na distribuição Debian já utilizada pelo grupo; e (*iii*) seu algoritmo de reconhecimento automático de hardware já estar bastante otimizado.

Quando executado desta maneira, o EXEHDA tem o Linux como sistema operacional hospedeiro. No momento da carga do software via CD, o administrador deve selecionar o perfil de ativação dos serviços naquele nodo em particular, uma das alternativas corresponde ao perfil de uma EXEHDAbase.

Tanto os perfis de ativação dos EXEHDAnodos, quanto os serviços do *middleware* podem ser atualizados a partir de uma BDA em particular. Nos procedimentos de inicialização está prevista uma consulta ao usuário neste sentido.

No capítulo seguinte estão apresentadas as principais aplicações desenvolvidas utilizando o EXEHDA, sendo caracterizados seus aspectos funcionais e adaptativos.

7 EXEHDA: APLICAÇÕES IMPLEMENTADAS

I hear and I forget.
I see and I remember.
I do and I understand.

- Chinese Proverb

Este capítulo tem por objetivo demonstrar o emprego dos mecanismos propostos para o EXEHDA no atendimento dos requisitos da Computação Pervasiva. As principais aplicações prototipadas com o uso do EXEHDA estão sumarizadas, e para a aplicação desenvolvida, denominada GeneAl, são detalhados aspectos relativos a seus fundamentos, comportamentos adaptativos e resultados atingidos.

7.1 Aplicação GeneAl

A aplicação GeneAl (***Gene****tic* ***Al****ignment*) é direcionada à resolução de um problema na área de biologia molecular classificado como "Alinhamento de Seqüências Genéticas", e foi submetido à publicação [SCH 2004a].

7.1.1 Caracterizando o problema de alinhamento de seqüências genéticas

A informação genética necessária para a realização das funções vitais de um organismo está contida em seu genoma. O genoma é composto por um determinado número de cromossomos, os quais são moléculas de ácido desoxirribonucléico (DNA) longas, construídas com a participação de proteínas. Por sua vez, genes correspondem a trechos de um cromossomo que são responsáveis pela transmissão das características hereditárias e desempenham um papel fundamental na síntese das enzimas em um ser vivo. Cada gene é composto por uma seqüência específica de nucleotídeos (adenina, citosina, guanina e timina).

O alinhamento de seqüências genéticas se refere à operação de comparação dos nucleotídeos que as compõem na busca por similaridades entre estas. São utilizadas bases de dados de bioseqüências conhecidas[2] e, através de um esquema de pontuação, são caracterizadas as seqüências que mais se assemelham. Entende-se por alinhamento ótimo aquele que maximiza esta pontuação obtida. Existem diversas instâncias desse problema. Particularmente, na aplicação GeneAl o objetivo consiste em comparar uma

[2] Existem diversas bases de dados genéticos disponíveis na esfera internacional, dentre estas o GenBank (NCCBI-NIH, EUA), o Protein Identification Resource (PIR, EUA), o DNA Data Bank of Japan (DDB, Japão) e o European Molecular Biology Laboratory (EMBL, Swiss).

seqüência específica de nucleotídeos com todas as bioseqüências, disponíveis em uma base de dados, identificando as que mais se assemelham com ela.

7.1.1.1 O algoritmo de alinhamento de seqüências genéticas

Na prototipação da aplicação GeneAl foi utilizada uma variante do algoritmo de Smith-Waterman para alinhamento de seqüências genéticas. Esta variante é direcionada à detecção de similaridades fracas entre seqüências separadas por uma significativa distância evolucionária [MEI 94, BUN 2000].

O procedimento de identificação das semelhanças entre duas seqüências emprega um sistema de pontuação. Neste procedimento, as seqüências a serem comparadas não precisam ter o mesmo comprimento, pois quando tiverem tamanhos diferentes é utilizado um procedimento de inserção de espaços de modo a igualá-las no comprimento. Considerando uma seqüência sobre a outra, a pontuação é determinada pela soma dos pontos atribuída a cada coluna, conforme o critério apresentado no quadro 7.1.

Evento	**Pesos**
Coluna com nucleotídeos iguais	1
Coluna com nucleotídeos diferentes	-1
Coluna contendo espaço	-2

Quadro 7.1: Regras para pontuação de alinhamento genético

Conjuntos de valores análogos aos apresentados no quadro 7.1 são usualmente empregados no alinhamento de seqüências genéticas. Estes valorizam a ocorrência de colunas com nucleotídeos iguais e penalizam a ocorrência de colunas que contenham nucleotídeos distintos ou espaços. O melhor alinhamento entre duas seqüências será aquele que obter melhor pontuação. A figura 7.1 apresenta o melhor alinhamento entre as seqüências S1 e S2. Neste exemplo, têm-se oito colunas com os mesmos nucleotídeos, uma coluna com nucleotídeos diferentes e uma coluna com espaço, ficando a pontuação do alinhamento igual a $5 \quad (8\times1+1\times(-1)+1\times(-2))$.

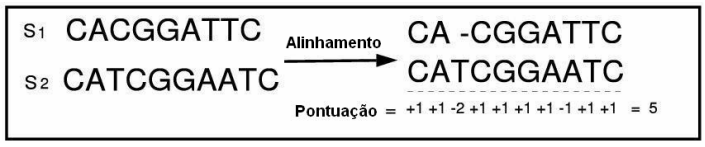


Figura 7.1: Alinhamento entre duas seqüências

O algoritmo utilizado na aplicação GeneAl para obter o alinhamento ótimo entre duas seqüências é divido em dois passos. O primeiro consiste do preenchimento de uma matriz combinando os dados das duas seqüências, e o segundo percorre essa matriz identificando o alinhamento com a maior pontuação.

Primeiro passo: preenchimento da matriz de dados

Considerando as seqüências S1 e S2, cujos comprimentos são M e N respectivamente, é construída uma matriz $A[M+1, N+1]$ empregando os seguintes critérios:

- a primeira linha e a primeira coluna (bordas superior e esquerda) são inicializadas com o valor -2 multiplicado pela posição da célula na matriz;
- as células restantes são calculadas segundo o critério:

$$A[i, j] = \max \begin{cases} A[i-1, j] - 2 \\ A[i-1, J-1] + P(i, j) \\ A[i, j-1] - 2 \end{cases}$$

onde:

$$P[i, j] = \begin{cases} +1 & se \quad S1[i] = S2[j] \\ -1 & se \quad S1[i] \neq S2[j] \end{cases}$$

Segundo passo: determinação do alinhamento ótimo

O alinhamento ótimo entre duas seqüências é obtido percorrendo a matriz A, partindo da célula $A[M, N]$ em direção à célula $A[0,0]$. Considerando a equação apresentada acima, o valor de uma célula $A[i, j]$ pode ter sido gerado a partir de três possibilidades:

- o valor da célula $A[i, j]$ foi determinado pela célula $A[i, j-1]$ (horizontal): o nucleotídeo em $S2[j]$ é alinhado com um espaço inserido em $S1[i]$;
- o valor da célula $A[i, j]$ foi determinado pela célula $A[i-1, j-1]$ (diagonal): o nucleotídeo em $S1[i]$ é alinhado com o nucleotídeo em $S2[j]$;
- o valor da célula $A[i, j]$ foi determinado pela célula $A[i-1, j]$ (vertical): o nucleotídeo em um nucleotídeo em $S1[j]$ é alinhado com um espaço inserido em $S2[i]$.

Uma vez determinado o alinhamento ótimo entre as seqüências S1 e S2, a pontuação correspondente é construída com os pesos apresentados no quadro 7.1.

7.1.2 Visão geral da organização da aplicação

A estratégia para tratar o problema de alinhamento de seqüências genéticas foi prototipada na aplicação GeneAl considerando um cenário no qual pesquisadores desejam encontrar seqüências similares a uma seqüência de interesse. A aplicação permite que este processo de busca seja realizado utilizando bases de dados de genes localizadas em diferentes instituições.

As duas principais razões para o processamento na GeneAl ser organizado de forma distribuída e concorrente são (*a*) a impossibilidade de migração das bases de bioseqüências da instituição hospedeira, seja por aspectos de propriedade intelectual seja por implicações decorrentes do tamanho dos arquivos que a compõem; (*b*) a elevada demanda de processamento empregada pelo algoritmo para alinhar todos os pares <*seqüência de entrada, seqüência base-de-dados[i]*>, onde a *seqüência base-de-dados[i]* representa cada uma das seqüências existentes em cada um dos bancos de dados de bioseqüências envolvidos.

7.1.2.1 Componentes da aplicação

A aplicação GeneAl foi concebida para uma operação distribuída e concorrente com uma hierarquia em três níveis. A aplicação é composta por entes de quatro naturezas, cujas funcionalidades são:

- GeneAl_GUI: ente de mais alto nível da aplicação que permite ao usuário parametrizar a execução e visualizar os resultados do processamento;
- GeneAl_Manager: ente responsável pela coordenação do processamento inter-celular;
- GeneAl_Master: ente responsável pela coordenação do processamento concorrente no âmbito de cada EXEHDAcel envolvida (intra-celular);
- GeneAl_Worker: ente responsável pela computação do algoritmo de alinhamento genético.

Um resumo das etapas de execução da aplicação GeneAl é apresentado na figura 7.2. Na etapa (a) está caracterizado o disparo da aplicação pelo usuário na EXEHDAcel aonde este se encontra, neste momento será clonado o ente GeneAl_GUI responsável pela interface da aplicação. Na etapa (b) o ente GeneAl_Manager é clonado pelo ente que implementa a interface da aplicação, e instanciado para execução. O GeneAl_Manager, por sua vez, clona e ativa na etapa (c) os entes GeneAl_Master cuja atribuição é clonar os entes que irão realizar o processamento, distribuir tarefas entre estes, e acumular os resultados produzidos pelos mesmos. Por fim, na etapa (d) os entes GeneAl_Worker que irão efetuar o processamento são então clonados e instanciados nos EXEHDAnodos da célula de execução.

Na figura 7.2, a direção das setas indica a organização lógica dos entes, do ente-filho para o ente-pai (relacionamento de composição). Note que a interação entre os entes se faz através da escrita/leitura na história do ente-pai.

Na seção 7.1.3 estão detalhados os comportamentos adaptativos correspondentes às etapas caracterizadas na figura 7.2.

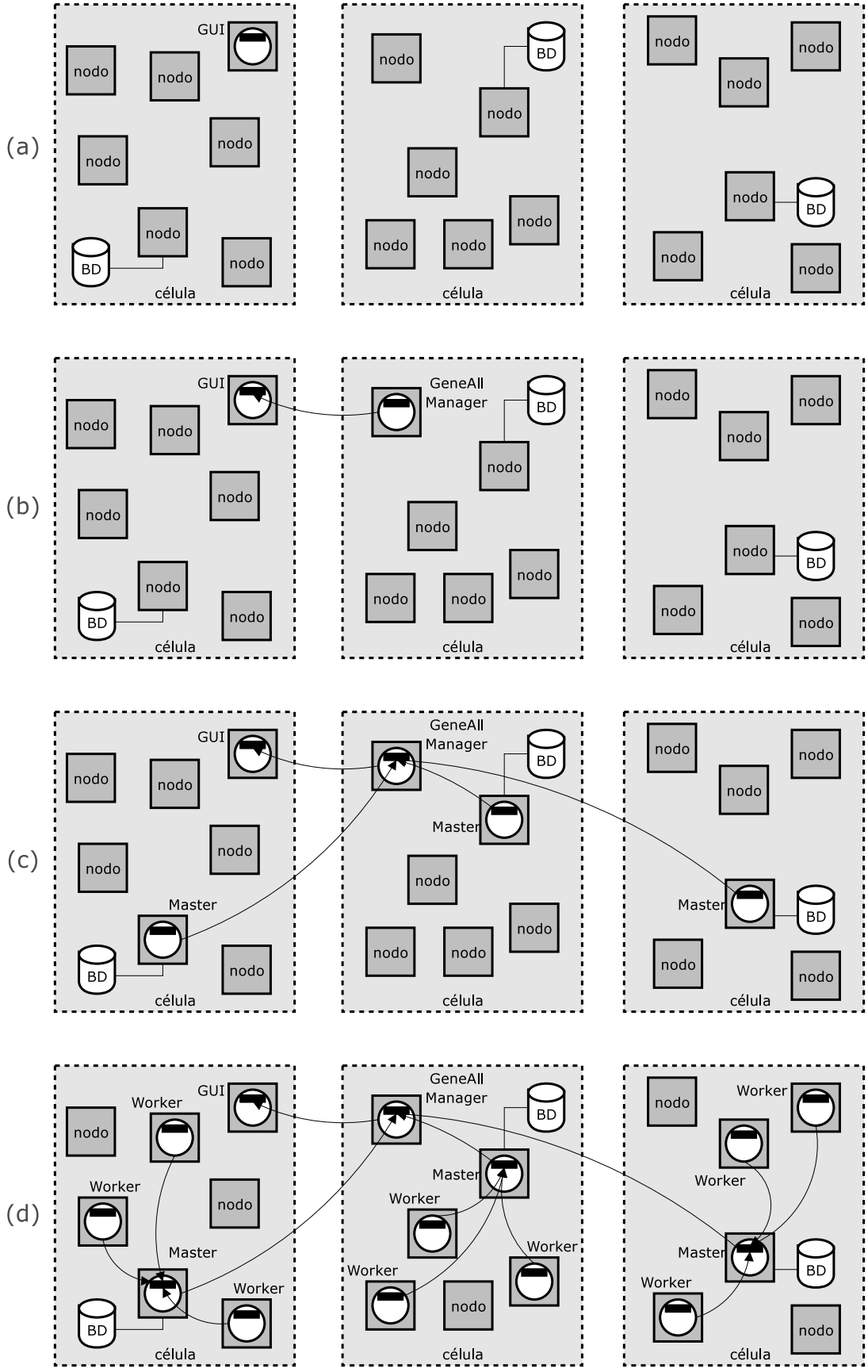


Figura 7.2: Etapas da aplicação GeneAl

7.1.2.2 Estratégias de particionamento de dados e geração de tarefas

Tarefas são compostas por um determinado número de bioseqüências provenientes das bases de dados. O tamanho das tarefas geradas pelos entes GeneAl_Master pode ser determinada de duas formas:

- **estática**: a quantidade exata de bioseqüências que irá compor as tarefas é especificada pelo usuário. Esta informação fica armazenada num arquivo de configuração;
- **dinâmica**: o tamanho das tarefas geradas é ajustado dinamicamente ao longo da execução de acordo com o contexto considerado: poder de processamento nominal e a taxa de ocupação corrente de cada nodo que abriga um ente GeneAl_Worker.

A abordagem dinâmica utiliza o conceito de *time-goal*. Neste conceito, as tarefas enviadas para cada GeneAl_Worker são redimensionadas dinamicamente em função do desempenho que ele obteve no processamento da tarefa anterior. O tamanho da primeira tarefa é estimado pelo usuário; o tamanho das tarefas seguintes é determinado pela relação entre o tempo de execução esperado (*time-goal*) e o tempo efetivamente gasto no processamento da tarefa imediatamente anterior. No caso do tempo de processamento praticado ter sido maior que o *time-goal* especificado, a próxima tarefa será menor que anterior; por outro lado, se o tempo de processamento praticado for menor que o *time-goal* o tamanho da tarefa seguinte será maior que o da anterior. Desta forma a adaptação é realizada em relação às características de hardware e também de acordo com flutuações na taxa de ocupação dos nodos que integram a execução.

7.1.3 Dinâmica operacional e comportamentos adaptativos ao contexto

O objetivo desta seção é associar os principais aspectos operacionais da aplicação GeneAl aos seus respectivos comportamentos adaptativos. Os comportamentos de adaptação contemplados são: (*a*) adaptação na carga da interface, (*b*) localização das bases de bioseqüências e (*c*) instanciação adaptativa dos componentes da aplicação. Esses comportamentos estão resumidos a seguir.

A) Adaptação na carga da interface:

O tipo de equipamento utilizado para disparo e acompanhamento da execução da aplicação constitui uma primeira estratégia de adaptação. Este comportamento é controlado pelos serviços *Executor* e *AdaptEngine*. A adaptação pertinente a carga da interface é do tipo funcional (vide seção 5.2.4). Neste sentido, a interface da aplicação foi disponibilizada através de um ente adaptativo cujo elemento contexto de interesse é o tipo de dispositivo que está sendo empregado (desktop. PDA). Foram construídos dois adaptadores (*adapters)* para implementar a funcionalidade do ente: (*i*) um, destinado a equipamentos de mesa (desktop) que utilizam a plataforma J2SE (vide figura 7.3); e (*ii*) outro, destinado a equipamentos portáteis (PDAs) que utilizam a plataforma J2ME (vide figura 7.4 e figura 7.5).

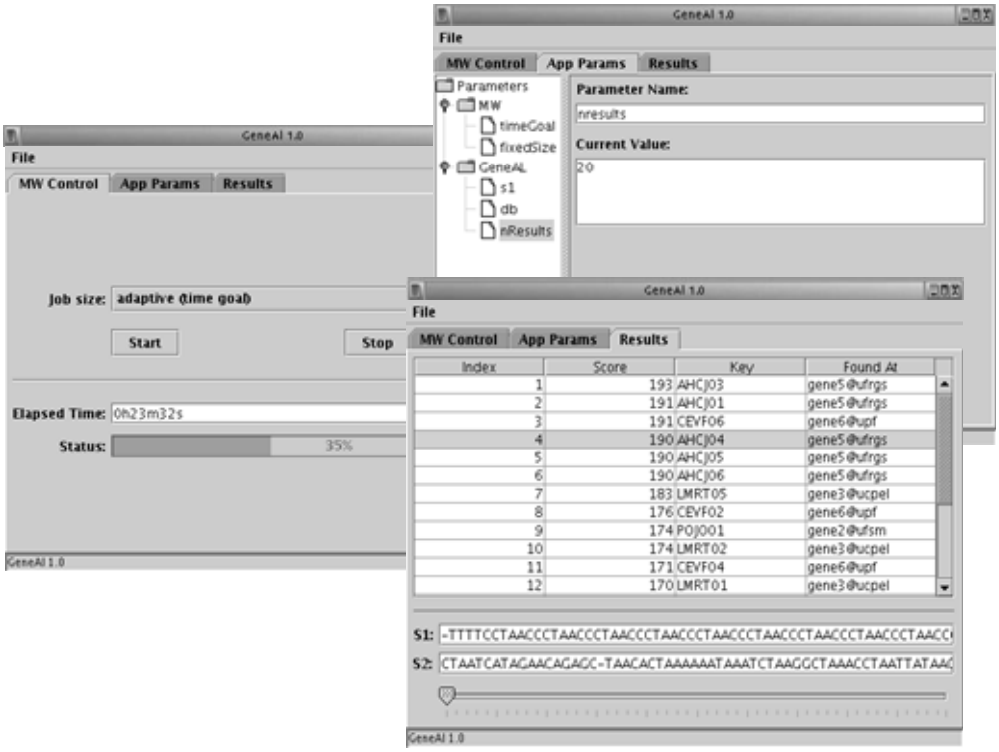


Figura 7.3: Interface da aplicação GeneAl para desktops

Dois procedimentos são realizados quando do disparo da aplicação GeneAl: (*a*) a definição dos parâmetros da execução; (*b*) a seleção das estratégias para particionamento dos dados. A seleção da estratégia de particionamento dos dados (vide figura 7.4b) é um procedimento repetido para as diferentes situações a serem avaliadas (vide seção 7.1.2.2).

B) Localização das bases de bioseqüências:

A determinação dos bancos de dados a serem utilizados no processamento é um procedimento centrado no serviço *Discoverer* do EXEHDA, e cujo objetivo é localizar dinamicamente as bases de bioseqüências no ISAMpe. O procedimento de descoberta é disparado quando a aplicação inicia, e o seu resultado reflete-se no valor do parâmetro GeneAl.db apresentado ao usuário (vide figura 7.4a). Através da modificação deste parâmetro, o usuário pode escolher as bases a serem utilizadas para uma execução em particular. Observe-se que as bases de bioseqüências são cadastradas de forma análoga aos recursos de hardware pelos administradores das células onde ficam alojadas;

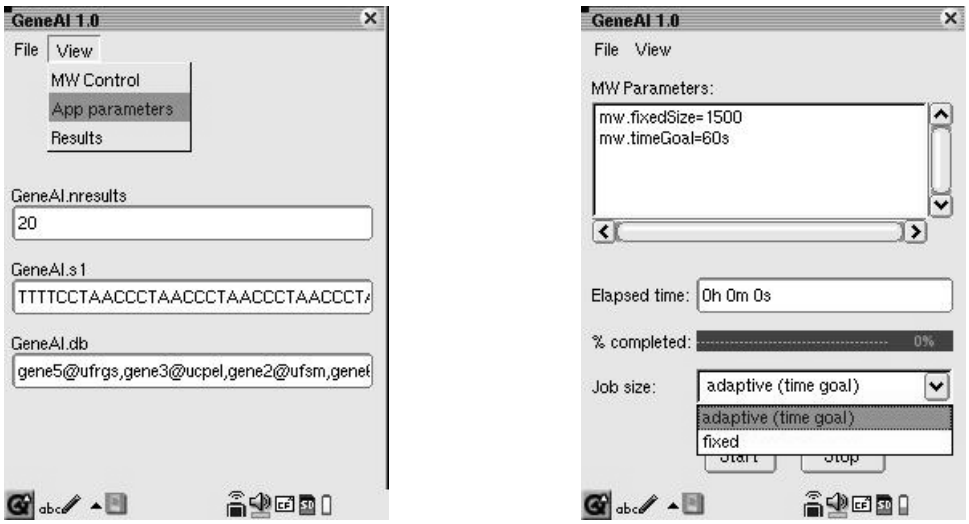
C) Instanciação adaptativa dos componentes da aplicação:

Ancorado a cada base de genes, é criado um ente GeneAl_Master, o qual é responsável pelo particionamento dos dados segundo a estratégia selecionada pelo usuário e pela criação dos entes GeneAl_Worker, que irão tomar parte no processamento.

O mecanismo básico para suporte a esta estratégia adaptativa é a instanciação de código sob demanda. Tanto a criação dos entes GeneAl_Master como a dos GeneAl_Workers é adaptativa no sentido que o nodo que irá receber a instanciação é selecionado dinamicamente pelo serviço *Scheduler* do EXEHDA, tendo como critério a

sua disponibilidade e capacidade de processamento. Ressalte-se que estes critérios traduzem a visão do desenvolvedor da aplicação. Como a aplicação foi executada com a premissa de ocupar todos os processadores disponíveis na célula, o foco da contribuição do serviço *Scheduler* se deu através da identificação dos processadores fisicamente disponíveis, e da garantia de alocação de uma única tarefa (GeneAl_Worker) a cada processador.

Os GeneAl_Workers consomem as tarefas disponibilizadas pelo GeneAl_Master de cada célula. As tarefas são tuplas escritas pelo ente GeneAl_Master em sua história, a qual pode ser lida pelos entes-filhos (GeneAl_Workers). Assim, outra dimensão de adaptação a cargo da aplicação é a determinação dinâmica do tamanho da tarefa a ser trabalhada (vide critério *time-goal*, seção 7.1.2.2), o qual de forma colaborativa com os outros mecanismos do *middleware*, por exemplo o escalonador, busca o melhor desempenho possível para a execução.

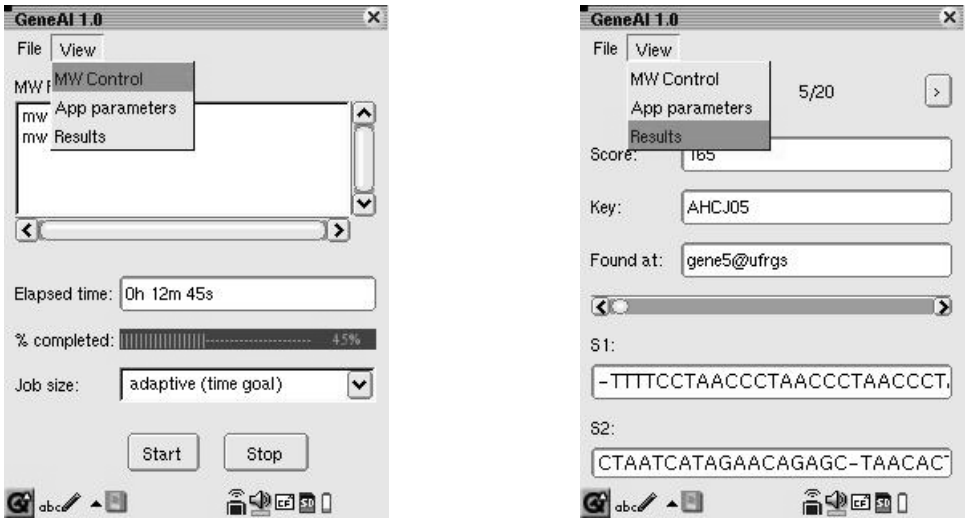


(a) definição de parâmetros

(b) seleção de estratégias para particionamento da carga

Figura 7.4: Interface da aplicação GeneAl para o PDA Zaurus 5600

Ao longo da execução da aplicação, entes do tipo GeneAl_Master aglutinam em nível celular os resultados parciais produzidos pelos GeneAl_Workers. Uma vez finalizado o processamento na célula, o GeneAl_Master publica o resultado na história do GeneAl_Manager, o qual combina este com os outros resultados parciais. O resultado final é disponibilizado na interface da aplicação (vide figura 7.5b).



(a) acompanhamento da execução (b) resultados atingidos

Figura 7.5: Resultados da aplicação GeneAl no PDA Zaurus 5600

7.1.4 Condições para realização dos testes

Os experimentos foram realizados utilizando equipamentos geograficamente distribuídos, com localização em municípios diferentes. Foram envolvidas quatro universidades, cada uma delas constituindo uma célula de execução (EXEHDAcel). Os recursos computacionais disponíveis em cada célula estão relacionados no quadro 7.2.

Quando necessária, a comunicação entre as células de execução ocorreu através do segmento da Internet denominado “Rede Tchê”, parte da RNP atuante no Rio Grande do Sul [RNP 2004].

EXEHDAcel	**Nodos**	**Rede**
UFRGS Universidade Federal do Rio Grande do Sul	2 Pentium IV 2.4 GHz 2 Pentium III 1 GHz 1 Atlhon 1 GHz 1 UltraSparc IIi 333 MHz	100 Mbps
UPF Universidade de Passo Fundo	6 Athlon 1 GHz	100 Mbps
UFSM Universidade Federal de Santa Maria	6 Pentium IV 2.4 GHz	100 Mbps
UCPel Universidade Católica de Pelotas	6 Pentium III 800 MHz	10 Mbps

Quadro 7.2: Recursos utilizados na aplicação GeneAl

Em concordância com os pressupostos apresentados na seção 7.1.2, cada EXEHDAcel envolvida no processamento dispõe de sua própria base de bioseqüências. As bases de dados têm tamanho médio de 46 MB, totalizando 184 MB de dados, o que representa aproximadamente 435.000 bioseqüências a serem analisadas. Os dados utilizados nas execuções foram obtidas no GenBank (NCCBI-NIH, EUA, <http://www.ncbi.nlm.nih.gov/genome/seq/>).

7.1.4.1 Instalação e ativação do EXEHDA

Para realização do experimento foi disponibilizado às instituições participantes um live-CD (vide seção 6.3) contendo toda plataforma de software necessária à ativação do EXEHDA. Com isto foram atingidos os objetivos de (*i*) desobrigar as instituições de instalarem e configurarem nos seus equipamentos o software correspondente ao EXEHDA e (*ii*) de minimizar os esforços de disparo e parametrização da execução da aplicação GeneAl.

Assim organizado, no processamento da aplicação GeneAl foi empregado o Linux como sistema operacional hospedeiro do EXEHDA, no caso o Knoppix versão 3.2 [KNO 2003]. O administrador da célula (usuário), quando da inicialização do *middleware*, tem a liberdade de informar o perfil de operação do nodo sendo ativado (EXEHDAbase, nodo de processamento, etc.). O live-CD é distribuído com o núcleo mínimo do EXEHDA (vide seção 4.4), e sob demanda são carregados os outros serviços necessários à execução da aplicação.

O live-CD prevê alternativa para que o próprio núcleo mínimo do *middleware* possa ser carregado através da rede no momento da sua inicialização (*bootstrap*), isto faculta que as execuções sejam realizadas com a última versão estável do EXEHDA.

7.1.4.2 Dimensionamento das tarefas

O problema do alinhamento de seqüências genéticas emprega um volume significativo de informações, e a forma como os dados são particionados tem forte repercussão no comportamento da aplicação GeneAl. As células de execução, por sua vez, se caracterizam por diferentes cenários no tocante aos recursos que disponibilizam. Para cada um destes cenários foram avaliadas três estratégias de particionamento de dados, as quais definem o volume dos dados enviados para os GeneAl_Workers:

- tarefas estáticas/pequenas: 200 bioseqüências são enviadas de cada vez;
- tarefas estáticas/grandes: 15.000 bioseqüências são enviadas de cada vez;
- tarefas dinâmicas: utilizam uma meta de tempo de processamento (*time-goal*) de 60 s para cada tarefa. O tamanho da tarefa é determinado dinamicamente durante a execução em função do poder de processamento disponível nos nodos.

O emprego de tarefas estáticas/grandes tem como objetivo observar o custo introduzido pela barreira de sincronização ao final do processamento. A heterogeneidade dos recursos em uma EXEHDAcel pode levar a uma situação em que, ao final da execução, nodos mais lentos permaneçam processando enquanto que outros mais poderosos estarão inativos.

Com o emprego das tarefas estáticas/pequenas o foco é observar o custo introduzido pelas comunicações no processamento, considerando que tarefas menores irão exigir um número maior de mensagens para que a execução seja concluída.

Por fim, o uso de tarefas com tamanho dinamicamente definido tem por objetivo aumentar o nível de paralelismo, seja pela diminuição da troca de mensagens, seja pela redução do tempo total de inatividade dos processadores quando do término da

aplicação. Observe-se que o poder de processamento disponível é conseqüência da capacidade de processamento e da ocupação do nodo no período.

As execuções da aplicação GeneAl foram disparadas do ISAM Desktop do usuário eTester (vide seção 6.2.3), a partir da EXEHDAcel localizada no Instituto de Informática da UFRGS. Foram realizadas três baterias distintas de testes, descritas nas próximas seções.

7.1.5 Experimento 1 - diferentes cenários de execução utilizando recursos dedicados

Este teste apresenta a execução da aplicação GeneAl em diferentes cenários, explorando a relação entre as estratégias de particionamento de dados e as características de processador e rede de cada célula. Os nodos de processamento ficaram dedicados à aplicação GeneAl durante todo período de testes. Os resultados atingidos estão apresentados na figura 7.6.

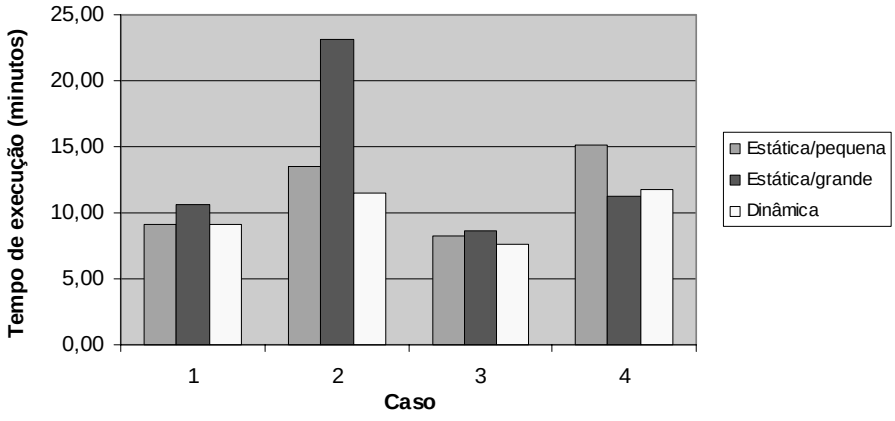


Figura 7.6: Estratégias de particionamento em diferentes cenários utilizando recursos dedicados

Para o processamento realizado na célula de execução existente na UPF, a qual é caracterizada por uma homogeneidade de hardware e por uma rede de interconexão Fast Ethernet (largura de banda nominal de 100 Mbps), as estratégias estática/pequena e dinâmica obtiveram praticamente o mesmo resultado. A estratégia de particionamento do tipo estática/grande, a despeito da homogeneidade do hardware envolvido registrou o pior desempenho, isto ocorreu devido a uma perda no nível de paralelismo no momento da finalização da execução, uma vez que o número de tarefas geradas não foi múltiplo do número de processadores na EXEHDAcel. Estes resultados podem ser vistos na figura 7.6, caso 1.

A célula de execução existente na UFRGS é caracterizada por heterogeneidade de hardware e rede de interconexão Fast Ethernet (100 Mbps). Sob estas condições o particionamento dinâmico das tarefas obteve melhor desempenho em relação às duas estratégias estáticas. A estratégia de particionamento em tarefas estáticas/pequenas não foi muito penalizada em função da velocidade da rede Fast Ethernet, por sua vez, a

estratégia estática/grande devido à elevada disparidade no poder de processamento dos nodos, teve significativo atraso no momento da finalização do processamento (vide argumentação apresentada no item 7.1.4.2). Os resultados correspondes a esta EXEHDAcel estão na figura 7.6, caso 2.

A célula de execução na UFSM apresentou um comportamento similar ao observado na UPF para todas as estratégias de particionamento dos dados (vide figura 7.6, caso 3). Ressalte-se que também na UFSM o número de tarefas estáticas/grandes não foi múltiplo do número de nodos.

Por fim, tem-se a célula de execução da UCPel, a qual é formada por nodos homogêneos e uma rede de interconexão Ethernet (10 Mbps). Nesta célula, o número de tarefas produzido pela estratégia de particionamento estática/grande foi proporcional ao de nodos, o que, somado ao aspecto da homogeneidade de hardware, permitiu paralelismo durante todo o período de processamento. Isto fez com que esta estratégia obtivesse o melhor desempenho. Por outro lado, a latência introduzida pela rede Ethernet fez com que a estratégia de particionamento estática/pequena e a estratégia dinâmica ficassem com o pior desempenho (vide figura 7.6, caso 4).

7.1.6 Experimento 2 - execução sob diferentes níveis de ocupação dos EXEHDAnodos

O objetivo deste teste foi avaliar o comportamento da aplicação GeneAl na situação em que os recursos não são alocados de forma exclusiva para uma aplicação ISAM em particular, mas os EXEHDAnodos alocados têm seu poder computacional em regime diferenciado de ocupação.

Foram utilizados nesta execução os recursos localizados na EXEHDAcel da UCPel (vide quadro 7.2). Os equipamentos nesta célula de execução são homogêneos. A condição de heterogeneidade na ocupação dos processadores foi obtida com o uso de um gerador sintético de carga computacional. O cenário dos recursos de processamento ficou com a configuração apresentada no quadro 7.3.

Equipamento	Quantidade	Ocupação
Pentium III, 800 MHz, 128 Mb	Dois nodos	60%
Pentium III, 800 MHz, 128 Mb	Dois nodos	30%
Pentium III, 800 MHz, 128 Mb	Dois nodos	0%

Quadro 7.3: EXEHDAnodos sob diferentes regimes de ocupação

Os valores apresentados na figura 7.7 mostram que no caso das tarefas de tamanho estático/grande, a significativa diferença no nível de ocupação dos nodos levou a uma perda de paralelismo quando do encerramento da aplicação. Por sua vez, a baixa banda-passante e a elevada latência da rede de interconexões (segmento de rede Ethernet a 10 Mbps) fez da estratégia estática/pequena a de pior desempenho. Por fim, a estratégia dinâmica na definição do tamanho das tarefas atingiu resultados melhores. Isto ocorreu porque a estratégia dinâmica permitiu um melhor equilíbrio entre o custo pertinente à barreira de sincronização ao final do processamento, e aquele introduzido pelo tráfego de mensagens.

O gerador de carga sintético utilizado na obtenção das condições de ocupação dos nodos apresentado no quadro 7.3 tem sua operação baseada em ciclos de ativação e desativação. Quando ativo, ele gera operações de ponto flutuante que ocupam todo poder de processamento disponível; quando inativo, permanece em total repouso. A partir daí, o controle do nível de ocupação do processador é feito através do ajuste do tempo médio de repouso em relação ao tempo médio de ativação a cada ciclo. Este gerador foi desenvolvido quando do trabalho [REA 2004].

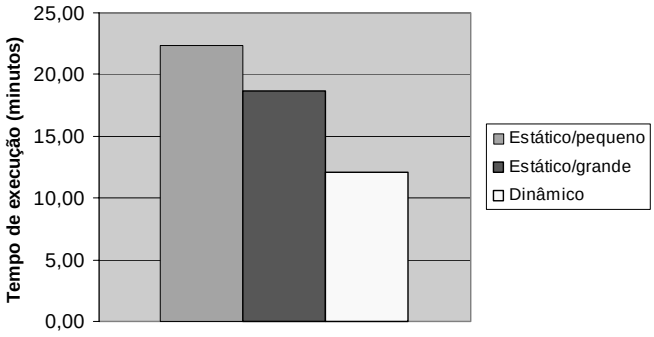


Figura 7.7: Resultados sobre diferentes regimes de ocupação dos EXEHDAnodos

A operação do gerador de carga é regida por um arquivo de controle denominado *AtivGen*. Este arquivo define a carga a ser gerada, especificando para cada ciclo os tempos de ativação e repouso, e a própria duração do ciclo. O arquivo *AtivGen* é produzido pelo programa *LoadGen.* Este programa produz a seqüência de ativação/desativação baseada em uma distribuição de probabilidades exponencial [REA 2004].

A estratégia de empregar um arquivo de controle (*AtivGen*) foi indispensável para que os testes desta bateria pudessem ser repetidos com os nodos da EXEHDAcel sob as mesmas condições de contexto.

7.1.7 Experimento 3 - diferentes estratégias de particionamento de dados e a escalabilidade

Por fim, este teste tem por objetivo mostrar o comportamento do desempenho para as três estratégias de particionamento de dados em função do número de processadores utilizados. O processamento foi realizado na célula de execução da UCPel.

Os resultados apresentados na figura 7.8 mostram a estratégia de particionamento estática/grande apresentando boa escalabilidade na maioria dos casos, exceto para aquelas situações em que o total de tarefas não foi múltiplo do número de processadores. No caso, o número de tarefas foi adotado como múltiplo de 6, isto fez com que o *speedup* para 4 e 5 processadores não apresentasse melhorias em relação ao obtido com 3 processadores.

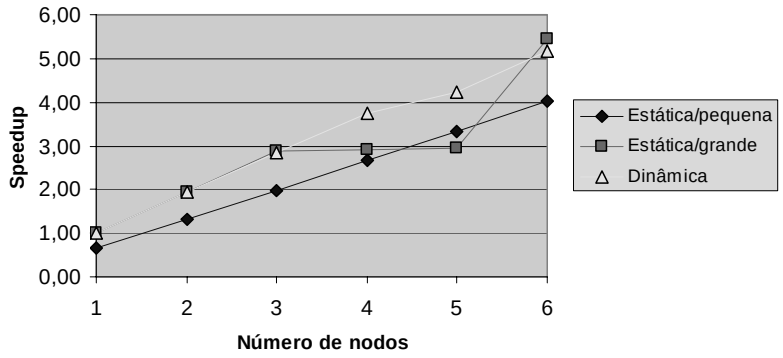


Figura 7.8: Estratégias de particionamento dos dados e seus ganhos de desempenho

A elevada latência da rede Ethernet fez com que o speedup para a estratégia estática/pequena ficasse menor que a observada nas outras. Por fim, a estratégia de determinação dinâmica do tamanho das tarefas foi a que obteve melhor linearidade no *speedup*. A estratégia de particionamento dinâmica é particularmente oportuna para situações em que não é possível saber de antemão a relação entre o volume de dados existente e os recursos que vão estar efetivamente disponíveis, situação comum na perspectiva da Computação Pervasiva.

7.2 Outras aplicações implementadas

Nesta seção são apresentadas outras aplicações que empregaram o EXEHDA. O critério para sua seleção foi terem sido objeto de publicação.

7.2.1 TicTac::Brutus – aplicação distribuída para análise de atraso de Circuitos VLSI

Determinar o atraso de circuitos VLSI é um aspecto de elevada importância para o projeto de circuitos digitais. O desafio que se apresenta é determinar este atraso com precisão significativa em um tempo aceitável. A exaustiva simulação numérica praticada pela aplicação TicTac::Brutus é caracterizada por um elevado custo computacional. O emprego do EXEHDA faculta que recursos heterogêneos, que se encontram distribuídos no ISAMpe, possam cooperar no atendimento da demanda de poder de processamento. Foi utilizada a estratégia de clonar entes para processamento e instalá-los para execução à medida que acontecia a disponibilidade de processadores. Conforme apresentado em [BRU 2004, BRU 2003], com o uso do EXEHDA o tempo total exigido para processamento do TicTac::Brutus foi significativamente reduzido nas diferentes simulações praticadas, sendo atingida uma escalabilidade de 62 nodos de processamento.

Outras informações estão disponíveis em <http://www.inf.ufrgs.br/~tictac/BRUTUS/>.

7.2.2 Mangoparrot – aplicação distribuída para posicionamento de células em circuitos VLSI

O problema do posicionamento em circuitos VLSI consiste em diminuir o tamanho total das conexões entre as células, a partir da otimização da localização das mesmas dentro do circuito integrado (chip). O posicionamento influi no desempenho, no

consumo e no tamanho do circuito final prototipado. Encontrar o posicionamento ideal, independentemente da métrica de qualidade, é um problema de otimização combinatorial cuja complexidade é NP-difícil.

A aplicação Mangoparrot é baseada na meta-heurística GRASP (*Greedy Randomized Adaptive Search Procedure*). O objetivo central com o desenvolvimento desta aplicação foi implementar um procedimento GRASP paralelo. O desenvolvimento teve como premissas a otimização do desempenho e a elevada escalabilidade. O emprego do EXEHDA aconteceu no sentido de suprir a infra-estrutura para uma prototipação distribuída e concorrente da aplicação Mangoparrot. A estratégia adaptativa foi utilizada na gerência da heterogeneidade e para a agregação de processadores a qual acontecia à medida que estes se tornavam disponíveis. O meio físico era constituído por equipamentos com diferentes hardwares e sistemas operacionais: estações com processadores Intel e Sparc, as quais empregavam os sistemas operacionais Windows, Linux e Solaris [BRU 2004a].

7.2.3 aBrot e calcRay- perfis adaptativos para balanceamento de cargas no ISAM

Fractais são figuras geométricas complexas e caracterizadas por uma possibilidade de detalhamento infinito. Deste modo, uma determinada região pode ser focalizada sendo mantida nesta o mesmo nível de detalhamento de todo fractal. No fractal de Mandelbrot, um dos mais conhecidos, o cálculo é baseado na aplicação recursiva de uma determinada função para cada ponto da figura que está sendo gerada. A construção de um fractal pressupõe a aplicação desta função inúmera vezes, o que exige uma demanda significativa de poder de processamento. O uso da concorrência se mostra uma alternativa oportuna para atender essa demanda.

Um dos principais problemas enfrentados para a divisão da carga de trabalho, quando de uma execução distribuída e concorrente do fractal de Mandelbrot, é a irregularidade dos requisitos de processamento de cada região. Para superar esta dificuldade, e poder planejar uma divisão das tarefas, foi implementado na aplicação aBrot uma estratégia baseada na geração de perfil que faculte antever os custos computacionais associados às diferentes regiões do fractal.

Uma vez sendo possível definir os custos computacionais das diferentes regiões, é possível fazer um balanceamento de cargas no momento destas serem distribuídas aos processadores existentes no ISAMpe. Nesta perspectiva, processadores com maior disponibilidade de poder computacional são contemplados com cargas maiores. Esta adaptação na distribuição das cargas considera não só o poder nominal do nodo como também a sua taxa de ocupação. A aplicação aBrot foi executada de forma distribuída e concorrente utilizando o EXEHDA como *middleware* e obteve ganhos de desempenho significativos [FRA 2003].

O balanceamento de carga adaptativo baseado em perfis como estratégia de adaptação também foi explorado em outra situação. A aplicação alvo neste caso implementou um algoritmo de Ray Tracing [FRA 2004]. Neste caso, também foram constatados ganhos de desempenho apesar da elevada heterogeneidade dos recursos computacionais envolvidos.

7.2.4 WalkEd – editor para a Computação Pervasiva

A aplicação WalkEd [AUG 2003] é um protótipo simplificado de um editor de textos, cuja modelagem considera a visão particular de contexto definido pelas características do dispositivo móvel em uso (capacidade e display) e pelo estado da conexão de rede (`connected, disconnecting, disconnected`). Para responder à variação do contexto, as estratégias de adaptação projetadas se referem à: alteração no código de alguns componentes da aplicação (carga dinâmica de código), alteração na interface apresentada ao usuário, habilitação/desabilitação de tarefas, migração para nodos mais poderosos. Os comandos de adaptação utilizados são `disconnect, reconnect, clone, move e discovery`.

A aplicação inicia sua execução carregando o código correspondente à criação da interface gráfica do editor conforme o elemento de contexto `deviceType` em uso (primeira dimensão de adaptação). No decorrer da execução, serviços como verificação ortográfica e impressão podem ser requisitados pelo usuário. Tais requisições disparam a criação de entes, os quais podem vir a executar em dispositivos distintos daquele em que executa a interface gráfica. A aplicação, como um todo, adapta-se à condição de conexão e desconexão, dado que os entes que compõem o editor podem estar localizados fisicamente em nodos distintos do sistema distribuído, uma vez que estes são criados próximos aos recursos físicos que necessitam, tais como dicionários e impressora (segunda dimensão de adaptação).

A semântica '*siga-me*' está refletida no comportamento dos entes que implementam a verificação ortográfica e a impressão (terceira dimensão de adaptação). Estes entes estão ancorados pelo ente que implementa a interface gráfica e, por conseguinte, representa a posição atual do usuário da aplicação. Nessa dimensão de adaptação, uma migração do ente interface gráfica (ativada pelo menu `goto`) dispara uma re-locação dos entes ancorados de forma a minimizar custos de comunicação respeitando, porém, suas dependências de acesso a recursos externos. Uma descrição detalhada do WalkEd pode ser encontrada em [AUG2004].

O capítulo seguinte apresenta as considerações finais deste trabalho, nele estão registradas as conclusões obtidas, as principais realizações e a perspectiva de trabalhos futuros.

8 CONCLUSÃO E TRABALHOS FUTUROS

The first ninety percent of the task takes ten percent of the time,
and the last ten percent takes the other ninety percent.

- Ninety-ninety rule of project schedules

Este capítulo revisita os quesitos de pesquisa considerados no desenvolvimento deste trabalho, ressalta as principais contribuições da tese e caracteriza as oportunidades de trabalho na frente de pesquisa pertinente ao EXEHDA.

8.1 Conclusão

Computação Pervasiva está sendo considerado o novo paradigma computacional do século XXI. Este novo paradigma contempla a mobilidade física e lógica em escala global, e para tanto considera que o usuário poderá acessar seu ambiente computacional independente de localização, de tempo e de dispositivo.

Uma adoção direta da tecnologia dos *middlewares* para computação distribuída não se mostra adequada à Computação Pervasiva. Primeiro, as primitivas de interação, tais como transações distribuídas, requisições a objetos ou chamada remota de procedimento, pressupõem uma elevada largura de banda e conexão permanentemente disponível. Isso não é compatível com as características inerentes a um ambiente que contempla mobilidade lógica e física. Segundo, *middlewares* orientados a objetos, como CORBA e Java-RMI, suportam principalmente comunicação ponto-a-ponto, requerendo co-existência temporal entre cliente e servidor, além de não tratarem a desconexão dos dispositivos móveis. Novamente temos uma situação de conflito com o ambiente da Computação Pervasiva, que requer desacoplamento espacial e temporal nas comunicações. Terceiro, ambientes distribuídos assumem que o meio aonde ocorre a execução é estacionário, deste modo, contemplam localização fixa para cada nodo e serviços conhecidos. Esta visão contrasta com o cenário altamente flexível proposto pela Computação Pervasiva, onde a disponíbilidas e a posição dos nodos altera-se no tempo, e novos serviços podem estar disponíveis dependendo da localização, sendo passíveis de serem descobertos dinamicamente. Finalmente, a carga computacional para execução de *middlewares* tradicionais, de reconhecida excelência e utilização, como CICS da IBM, Tuxedo da BEA e Jini da SUN, é alta para que estes possam ser carregados e executados em nodos portáteis (PDAs em geral).

Outro aspecto importante na construção de soluções para Computação Pervasiva diz respeito ao equilíbrio entre transparência e consciência do contexto de interesse da aplicação. *Middlewares* tradicionais são construídos baseados em abordagens que enfatizam a transparência, onde programadores não necessitam conhecer o estado dos elementos computacionais associados aos recursos que alojam os serviços empregados

pelas aplicações. Enquanto que em sistemas distribuídos que não contemplam mobilidade física e lógica isto é possível e, muitas vezes desejável, esconder completamente as informações de contexto, por exemplo a localização, em ambientes com mobilidade se torna inadequado. Para oferecer transparência, os *middlewares* precisam tomar decisões em nome das aplicações, generalizando soluções e sacrificando a flexibilidade. No entanto, considerando as demandas introduzidas pelas aplicações na Computação Pervasiva, é mais eficiente que as decisões sobre utilização dos recursos levem em conta informações específicas da aplicação, do dispositivo utilizado, bem como as relativas ao estado do meio de execução corrente. Por outro lado, na Computação Pervasiva é essencial que o próprio *middleware* seja adaptativo ao meio, além de prover mecanismos para a adaptação das aplicações que executam sobre ele. Esta situação também diverge da premissa dos *middlewares* tradicionais de ocultarem aspectos relativos à consciência do contexto.

Para criação de aplicações pervasivas genéricas se faz necessária uma infra-estrutura de suporte para seu desenvolvimento e execução, até então inexistente. Este é o objetivo do projeto ISAM em desenvolvimento no II/UFRGS, desde 2001.

O *middleware* EXEHDA foi concebido com a premissa de disponibilizar um ambiente pervasivo, o qual foi denominado ISAMpe, e de dar suporte à execução de aplicações ISAMadapt neste ambiente. Para tal emprega a abstração Ambiente Virtual do Usuário, e disponibiliza o modelo computacional e os serviços necessários para que as aplicações possam implementar o estilo *siga-me* (*follow-me applications*).

Um aspecto significativo quando do uso da adaptação dinâmica é a relação entre flexibilidade e consumo de recursos. Na perspectiva do EXEHDA de considerar (*i*) uma rede de abrangência global, e (*ii*) a mobilidade física do usuário, com ou sem o seu equipamento, não há como saber a priori quais serviços do *middleware* os EXEHDAnodos poderão necessitar. Em outras palavras, é inviável antever qual será o escopo das mudanças quando o *middleware* necessitar ser reconfigurado, dada a diversidade de situações às quais os EXEHDAnodos poderão vir a se expor. Neste sentido, e considerando também que o ISAMpe prevê o uso de equipamentos de pequeno porte (PDAs), cuja memória disponível limita o número de componentes alternativos do *middleware* que podem ser carregados de antemão: uma das contribuições da proposta do EXEHDA é a superação desta situação pela combinação da estratégia adaptativa com dois paradigmas de mobilidade de código: carga de código sob demanda e instanciação remota de código.

O requisito de manter os nodos portáteis operacionais, mesmo durante os períodos de desconexão planejada, motivou, além da concepção de primitivas de comunicação adequadas à premissa de uma estratégia de desconexão controlada pelo usuário, a separação de serviços que implementam operações de natureza distribuída em duas instâncias: uma instância local ao EXEHDAnodo (nodal), e uma instância alojada na EXEHDAbase (celular). A instância nodal para suporte aos serviços distribuídos do *middleware* exibe adicionalmente à adaptação em tempo de carga, uma adaptação em função do estado de conectividade do dispositivo. Com isso, o dispositivo pode permanecer operacional, desde que, para satisfação de uma dada requisição, o acesso a um recurso externo ao dispositivo seja prescindível. Por outro lado, a instância celular, em execução na EXEHDAbase, provê um "ponto-de-referência" para realização de operações que requeiram algum tipo de coordenação distribuída. Observe-se que a EXEHDAbase é, por concepção, uma entidade estável dentro da célula, permitindo que

demais recursos tenham um caráter mais dinâmico no que se refere a sua disponibilidade efetiva no ISAMpe.

De forma análoga às aplicações de alto desempenho, uma aplicação na Computação Pervasiva, via de regra, deve oferecer resultados o mais rapidamente possível ao seu usuário. Isto é uma exigência implícita à situação de mobilidade: equipamento operando com bateria de pouca autonomia, custo de uso da rede global, inserção no tempo/espaço do contexto da tomada de decisão (e.g. reuniões com clientes), etc.

Por fim, a seleção de tecnologias para concepção de um *middleware* que materialize a premissa do Projeto ISAM de que a infra-estrutura de suporte à pervasividade em escala global pode ser construída através da integração de três áreas, Computação Móvel, Computação em Grade e Computação Consciente do Contexto, é uma tarefa desafiadora. Particularmente, a opção por Java se mostra promissora por simplificar o tratamento da heterogeneidade, além de prover funcionalidades básicas de segurança e operação distribuída, permitindo uma concentração dos esforços de pesquisa para atender os aspectos inovadores inerentes à Computação Pervasiva.

No Quadro 8.1, a seguir, é feita uma comparação entre os principais *middlewares* utilizados como referência para a definição deste trabalho, e a solução adotada no ISAM-EXEHDA. Salienta-se que os *middlewares* abordam aspectos específicos, sendo característica particular do EXEHDA a integração desses aspectos na concepção de seus serviços básicos, na busca de generalidade, reusabilidade e desempenho quando do projeto e da execução de aplicações pervasivas.

Aspecto da Pervasividade	**Outros *Middlewares***	**ISAM/EXEHDA**
Comunicação com desacoplamento temporal e espacial: suporte a mobilidade física e lógica.	**Lime e Tota**. Utilizam a plataforma de agentes móveis MARS, programada em Java. Agentes trafegam entre nodos móveis e interagem via um espaço de tuplas transientemente compartilhado. Agentes nos nodos operando localmente, utilizam o *middleware* para disseminar tuplas em uma rede ad-hoc [PIC 2001, MAM 2003].	A unidade de computação no nível do EXEHDA é o OX, o qual representa uma entidade migrável na qual podem ser externamente associados atributos (meta informações) de execução visíveis em todo o ISAMpe. O EXEHDA disponibiliza diferentes modelos de comunicação: troca de mensagens, chamadas remotas de método e espaços de tuplas. Ressalte-se que todos os modelos disponibilizados suportam a premissa da desconexão planejada.

Quadro 8.1: Comparação do EXEHDA com outros *middlewares (continua)*

Solução genérica para reconhe-cimento de contexto	**Context Toolkit** . *Framework* em Java para programação de aplicações. Fornece classes para programar sensores, agregadores, tradutores e notificadores. Contexto define elementos de um ambiente que detectam a presença de pessoas ou objetos e permitem reações a esta [DEY 2000]. **Solar**. Aborda a disseminação de informação de contexto em uma rede ad-hoc via um grafo de operadores (filtros, agregadores, transformadores e unificadores) que fazem a composição de informações [SOL 2002].	EXEHDA suporta a premissa de que contexto é toda a informação de interesse passível de ser obtida do ambiente, seja ela pertinente a recursos físicos ou a entidades lógicas. Como entidade lógica estão incluídos componentes da própria aplicação. Esta visão confere maior flexibilidade na personalização dos elementos e contexto de interesse do *middleware* e da aplicação. A informação de contexto é produzida, em alto nível, pelo serviço *ContextManager*, a partir da informação extraída pelo subsistema de monitoração, pelo uso de filtros denominados cadeias de detecção de contexto (context-chains), sendo que as alterações no contexto podem ser obtidas por inquisição direta ao serviço *ContextManager*, ou por notificação decorrente de uma política publish/subscribe, sendo esta última a forma preferencial (otimiza a disseminação).
Gerenciamento e descoberta de recursos e serviços	**JiniME**. Versão reduzida do protocolo usado em desktops. Cada dispositivo contém o mecanismo de busca embutido, e comunica-se com seus pares para encontrar o recurso [JME 2002]. **Solar.** Provê serviço de nomes que permite aos recursos anunciarem-se, e às aplicações fazerem consultas sensíveis ao contexto [SOL 2002]. **Colomba.** Aborda a associação recurso-agente face à migração do agente para outros nodos. Baseado na plataforma SOMA para agentes móveis, utiliza metadados para descrever os recursos, e serviços para mediar o acesso a estes recursos conforme políticas especificadas [BEL 2003].	Funcionalidades providas pelos serviços *Discoverer*, *ResourceBroker*, *Scheduler* e *CellInformationBase* contemplam a descoberta e alocação de recursos em um sistema cuja composição é dinâmica (recursos e serviços), heterogêneo, multi-institucional e largamente distribuído.

Quadro 8.1: Comparação do EXEHDA com outros *middlewares* (continua)

Ambiente Pervasivo Pró-ativo	**Aura.** Computação centrada no usuário. O *middleware* pró-ativamente controla as atividades do usuário e disponibiliza os recursos e serviços que este precisa, fazendo as adaptações necessárias [GAR 2002].	Este aspecto é abordado de outra forma. Não há uma centralização no perfil e atividades do usuário pois o EXEHDA é direcionado ao desenvolvimento de aplicações de propósito geral. Contudo, serviços como AVU, BDA, *SessionManager*, permitem ao acesso pervasivo por parte do usuário a seu ambiente computacional, ressaltando-se que esse acesso é adaptado às características dos recursos computacionais empregados no momento. Adicionalmente, o *middleware* pode atuar pró-ativamente na adaptação dos aspectos não funcionais das aplicações através de re-escalonamentos, os quais podem considerar, entre outros aspectos, o atendimento da semântica *siga-me* à medida que o usuário realiza deslocamentos.
Execução de Aplicações Legadas em um Ambiente Pervasivo.	**Gaia**. Usa a plataforma GaiaOS e CORBA. Define espaços ativos (sala de aula, laboratório, etc.) como ambiente pervasivo e fornece serviços para obter informações sobre este espaço. Fornece *framework* para disparar as aplicações correspondentes (adaptadas) ao espaço ativo [ROM 2002].	O EXEHDA considera um escopo de operação em escala global. Emprega a tecnologia Java como plataforma base. Os mecanismos existentes em Java podem ativar códigos legados. Os mecanismos do subsistema de adaptação pode ser empregados para definir dinamicamente a aplicação (ou versão) que será disparada em função do contexto.
Infra-estrutura para computação distribuída em larga escala - Computação em Grade.	**Globus**. Fornece ferramentas para distribuição de programas e dados (Gridftp). A operação é via de regra controlada pelo usuário e não são contemplados procedimentos de adaptação. Com base em procedimento de autenticação única (*Single Sign-On*) podem ser disparados comandos remotos. Unidade de concorrência são aplicações (processos) [FOS 2001].	Distribuição de dados e código (sem a participação do usuário, automatizada), gerenciada pelo *middleware* com base no contexto de execução da aplicação. É feita instalação tanto dos componentes da aplicação, como dos serviços do EXEHDA sob demanda, e de forma adaptada ao estado dos recursos disponíveis. A unidade de concorrência são os componentes da aplicação.

Quadro 8.1: Comparação do EXEHDA com outros *middlewares*

8.1.1 Contribuições da pesquisa

Cientificamente, esta tese contribui com a proposição de um *middleware* direcionado à Computação Pervasiva. Registre-se que para o ambiente pervasivo, não existiam até então, *middlewares* que atendessem suas necessidades da forma integrada como a proporcionada pelo EXEHDA. Somente soluções parciais estavam disponíveis,

tais como as providas pelos *middlewares* específicos para comunicação, adaptação ou obtenção de informações de contexto.

Outra importante contribuição é a integração do *middleware* com uma linguagem intencionalmente concebida para o desenvolvimento de aplicações da Computação Pervasiva. Esta integração, dentre outros aspectos, é oportuna para redução do caráter prioritariamente ad-hoc, existente nas aplicações atuais, potencializando a extensão e o reuso das soluções que oferecem.

A seguir estão destacados os principais resultados atingidos durante o desenrolar das atividades de pesquisa atinentes a esta tese de doutorado:

- identificação das principais pesquisas direcionadas à proposição de *middlewares* para a Computação Pervasiva. Um resumo deste estudo constitui o capítulo 2;
- construção dos requisitos que devem ser supridos por um *middleware* destinado ao cenário da Computação Pervasiva, tendo sido concebido um *middleware* que contempla tais requisitos. Os requisitos estão registrados no capítulo 3;
- concepção da abstração OX como elemento de computação no nível do *middleware*, bem como de serviços destinados à implementação de aplicações da Computação Pervasiva. O nível de abstração atingido se mostrou adequado enquanto permitiu que as construções da linguagem ISAMadapt pudessem ser mapeadas com relativa simplicidade;
- definição de uma organização escalável, adequada a uma composição de recursos de natureza multi-institucional, baseada na abstração de "célula de execução" (EXEHDAcel). Sob a perspectiva da dinamicidade de agregação de novos recursos, a definição da abstração EXEHDAbase é oportuna por simplificar o processo de gerência e as interações inter-celulares no ISAMpe;
- delineamento, no capítulo 6, de um modelo de gerência para o ISAMpe que considera o aspecto multi-institucional dos recursos envolvidos;
- especificação e implementação de uma ferramenta gráfica para execução dos procedimentos pertinentes ao gerenciamento do ISAMpe, denominada EXEHDA-AMI;
- materialização da arquitetura de software do ISAM, provendo uma infra-estrutura até então inexistente para o projeto e o suporte à execução de aplicações conscientes do contexto em um ambiente de Computação Pervasiva;
- viabilização da implementação de aplicações distribuídas, móveis e conscientes do contexto. As aplicações desenvolvidas encontram-se registradas no capítulo 7;
- participação na definição das linhas gerais do Projeto ISAM, desde a definição em mais alto nível da arquitetura de software, até a opção pelo modelo de adaptação colaborativa multinível;

- participação na submissão do projeto ISAM: Infra-estrutura de Suporte às Aplicações Móveis distribuídas no edital 03/2001-PROADI da FAPERGS. O mesmo foi aprovado para financiamento;
- participação na submissão do projeto contextS, um *Middleware* para o Desenvolvimento de Aplicações Sensíveis ao Contexto no edital conjunto do SEPIN/MCT-FINEP-CNPq 1/2002, aprovado para financiamento;
- participação na submissão do projeto Projeto GRADEp: *Middleware* para gerenciar um ambiente de grade pervasiva. O mesmo foi selecionado para constituir tema de grupo de trabalho na RNP. Início das atividades em agosto de 2004.
- abertura de alternativas para trabalhos de pesquisa no âmbito do GPPD. Algumas destas opções, conduzidas em trabalhos de mestrado, estão elencadas na seção 8.2;
- integração com grupo de pesquisa da comunidade internacional, tendo sido realizado estágio no Paderborn Center for Parallel Computing (PC2) da Univerdade de Parderborn na Alemanha. As atividades foram supervisionadas pelo Prof. Dr. Hans-Ulrich Heiss;
- divulgação dos resultados do trabalho realizado durante o desenvolvimento da tese através de publicações. As principais estão relacionadas na seção 8.1.2;
- difusão no âmbito das instituições de ensino superior do conhecimento pertinente à área de suporte à execução de aplicações na Computação Pervasiva. Neste sentido foi oferecida disciplina de projeto de pesquisa intitulada “Aplicações Móveis Distribuídas: Estudo e Integração de Mecanismos de Suporte à Execução”. Adicionalmente foram realizados nove trabalhos de conclusão de curso de graduação relacionados com a pesquisa desenvolvida no EXEHDA [GON 2004, MAR 2004, CER 2003, COS 2003a, FEH 2003, FRZ 2003, PER 2003, TAV 2002, REA 2001];
- repasse de tecnologia, materializado através do site do EXEHDA na Internet. Neste site <http://www.inf.ufrgs.br/~exehda> além de informações sobre o projeto, encontram-se disponibilizados, sob uma política Open-Source, o código fonte do *middleware* e de ferramentas auxiliares desenvolvidas.

8.1.2 Publicações

Nesta seção, estão relacionadas as publicações pertinentes ao Projeto ISAM/EXEHDA. Foram selecionadas aquelas que tiveram participação efetiva do autor. A ordem considera o ano de publicação.

Publicações 2001

WSCAD 2001. Adenauer Yamin, Iara Augustin, Jorge Barbosa, Cláudio Geyer, Explorando o Escalonamento no Desempenho de Aplicações Móveis Distribuídas, II *WORKSHOP EM SISTEMAS COMPUTACIONAIS DE ALTO DESEMPENHO*, **Anais...** Pirenópolis, Goiás, Brasil, set., 2001 [YAM 2001].

SBAC-PAD 2001. Jorge Barbosa; Adenauer Yamin; Patrícia Vargas; Débora Ferrari; Egon Schaeffer; Cláudio Geyer. Using Mobility and Blackboards to Support a Multiparadigm Model Oriented to Distributed Processing. In: SYMPOSIUM ON COMPUTER ARCHITECTURE AND HIGH PERFORMANCE COMPUTING, 13., 2001, Pirenópolis, Brasil. **Proceedings…** Brasília: UNB, September 2001. p.187-194 [BAR 2001].

CACIC 2001. Luciano da Silva, Adenauer Yamin, Iara Augustin, Jorge Barbosa, Cláudio Geyer, Mecanismos de Suporte ao Escalonamento em Sistemas com Objetos Distribuídos Java, VIII CACIC *CONGRESO ARGENTINO DE CIENCIAS DE LA COMPUTACIÓN*, **Anais...** Santa Cruz, Argentina, oct, 2001 [SIL 2001].

CACIC 2001. Edson Silva r, Adenauer Yamin, Iara Augustin, Jorge Barbosa, Cláudio Geyer, Hierarquia de Gerenciamento de Redes com Componentes Móveis, VIII CACIC *CONGRESO ARGENTINO DE CIENCIAS DE LA COMPUTACIÓN*, **Anais...** Santa Cruz, Argentina, oct, 2001 [SVA 2001].

CACIC 2001. Iara Augustin, Adenauer Yamin, Jorge Barbosa, Cláudio Geyer, Requisitos para o Projeto de Aplicações Móveis Distribuídas, VIII CACIC *CONGRESO ARGENTINO DE CIENCIAS DE LA COMPUTACIÓN*., **Anais...** Argentina, oct, 2001 [AUG 2001].

JCC 2001. Edvar Araújo, Adenauer Yamin, Iara Augustin, Luciano Silva, Cláudio Geyer, Uma Proposta de Monitoração para Visualização de Aplicações Distribuídas Java, JORNADAS CHILENAS DE COMPUTACIÓN 2001 - *V WORKSHOP EN SISTEMAS DISTRIBUÍDOS Y PARALELISMO*, **Anais...** Chile. 5-9 Nov. , 2001 [ARA 2001].

RITA 2001. Iara Augustin, Adenauer Yamin, Edson Nascimento Jr, Jorge Barbosa,Gerson Cavalheiro, Cláudio Geyer, ISAM: um *Middleware* para Aplicações Móveis Distribuídas**. RITA – Revista de Informática Teórica e Aplicada**. Edição Especial – Sistemas Operacionais. Vol. VIII n. 2, 2001 [AUG 2001a].

Publicações 2002

NET-CON02. Adenauer Yamin, Iara Augustin, Jorge Barbosa, Cláudio F.R. Geyer. ISAM: a Pervasive View. In Distributed Mobile Computing. Network Control and Engineering for QoS, Security and Mobility with focus on Policy-based Networking (IFIP and IEEE Conference). **Proceedings...** Paris, France, 21-25 oct. 2002 [YAM 2002].

SBAC-PAD 2002. Adenauer Yamin, Iara Augustin, Jorge Barbosa, Luciano C. da Silva, Rodrigo A. Real, Gerson Cavalheiro, Cláudio F.R. Geyer. A Framework for Exploiting Adaptation in High Heterogeneous Distributed Processing. 14th IEEE Symposium on Computer Architecture and High Performance Computing. **Proceedings…** Vitória - Brazil, October 28-30 [YAM 2002a].

SCCC 02. Adenauer Yamin, Iara Augustin, Jorge Barbosa, Luciano Cavalheiro da Silva, Gerson H. Cavalheiro, Cláudio F.R. Geyer. Collaborative Multilevel Adaptation in Distributed Mobile Applications. 12th IEEE International Conference of the Chilean Computer Science Society, **Proceedings…** Atacama, CHILE, 6-8 novembro, 2002 [YAM 2002b].

PDCN 2002. Iara Augustin, Adenauer Yamin, Jorge Barbosa, Cláudio Geyer, Towards Taxonomy for Mobile Applications with Adaptive Behavior. International Symposium on Parallel and Distributed Computing and Networks (PDCN 2002). **Proceedings…** Innsbruck, Austria. 18-21/feb, 2002 [AUG 2002].

CATA 2002. Iara Augustin, Adenauer Yamin, Cláudio Geyer, Distributed Mobile Applications with Dynamic Adaptive Behavior. 17th International Conference on Computer and their Applications(CATA 2002). **Proceedings…** San Francisco, CA. 4-6/april, 2002, R. Gantenbein and S. Shin Editors, ISCA Publishing, ISBN 1-880843-42-0, p.372-375 [AUG 2002a].

ISCC 2002. Iara Augustin, Adenauer Yamin, Jorge Barbosa, Cláudio Geyer, ISAM - a Software Architecture for Adaptive and Distributed Mobile Applications, 7th IEEE Symposium on Computers and Communications, Taormina, **Proceedings…** Italy, 1-4/july, 2002 [AUG 2002b].

ICPADS 2002. Jorge Barbosa; Adenauer Yamin; Patrícia Vargas; Cláudio Geyer. Holoparadigm: a Multiparadigm Model Oriented to Development of Distributed Systems In. INTERNATIONAL CONFERENCE ON PARALLEL AND DISTRIBUTED SYSTEMS (ICPADS 2002), 2002, Jung-Li City **Proceedings…** New York: IEEE Press, 2002 [BAR 2002a].

IDPT 2002. Rodrigo Reis, Carla Reis, Iara Augustin, Adenauer Yamin, Daltro Nunes, Cláudio Geyer, Towards a Software Process Model to Support the Design of Mobile Computing Applications, 6th

World Conference on Integrated Design and Process Technology, **Procedings…** Pasadena California, USA, June [REI 2002].

Publicações 2003

JHPCA 2003. Adenauer Yamin, Iara Augustin, Jorge Barbosa, Luciano C. da Silva, Rodrigo A. Real, Gerson Cavalheiro, Cláudio F.R. Geyer. **Towards Merging Context-aware, Mobile and Grid Computing.** IN INTERNATIONAL JOURNAL OF HIGH PERFORMANCE COMPUTING APPLICATIONS. London: Sage Publications. v.17, n.2, p.191-203, June 2003 [YAM 2003].

Mobile Computing Handbook. Iara Augustin, Adenauer Yamin, Luciano C. Silva, Rodrigo Real, Gustavo Frainer, Gerson Cavalheiro, Claudio Geyer**. ISAM, joing context-awareness and mobility to building pervasive applications.** I. Mahgoub and M. Ilyas Ed. Florida. CRC Press. **(book chapter** to be published at april, 2003) [AUG 2003].

IADIS 2003. Rodrigo Real, Adenauer Yamin, Iara Augustin, Jorge Barbosa, Luciano da Silva, Gustavo Frainer, Cláudio Geyer. Tratamento da incerteza no escalonamento de recursos em Computação Pervasiva. In: CONFERÊNCIA IADIS IBERO-AMERICANA WWW/INTERNET, Nov, 2003, Algarve, Portugal. **Anais...** Algarve: IADIS press, Novembro 2003. p.167-170 [REA 2003].

GRID 2003. Rodrigo Real, Adenauer Yamin, Iara Augustin, Jorge Barbosa, Luciano da Silva, Gustavo Frainer, Cláudio Geyer. Resource scheduling on grid: handling uncertainty. IEEE/ACM 4th INTERNATIONAL WORKSHOP ON GRID COMPUTING, Nov, 2003, Phoenix, Arizona. **Proceedings…** New York: IEEE Press, November 2003 [REA 2003a].

WSGPPD 2003. Adenauer Yamin, Iara Augustin, Jorge Barbosa, Luciano C. da Silva, Rodrigo A. Real, Cláudio F.R. Geyer. **EXEHDA: Um Ambiente de Execução para Adaptação Dinâmica ao Contexto de Aplicações na** Computação Pervasiva. Cadernos de Informática. Porto Alegre: PPGC/UFRGS. V. 3, n. 1, p.115-120, junho de 2003 [YAM 2003a].

ERAD 2003. Dario Fernandes Franz; Marcelo Augusto Cardozo Junior; Jorge Luis Victória Barbosa; Adenauer Correa Yamin; Cláudio Fernando Resin Geyer. EXEHDA-CC x JADA: Uma análise de espaço de objetos compartilhados em Java. In: ESCOLA REGIONAL DE ALTO DESEMPENHO (ERAD 2003), 2003, Santa Maria. **Anais...** . Santa Maria: SBC/UFSM, 2003. p. 177-180 [FRZ 2003].

ERAD 2003. Fernando Luis Caprio da Costa Junior; Jorge Luis Victória Barbosa; Iara Augustin; Adenauer Corrêa Yamin; Cláudio Fernando Resin Geyer. Uso do Network Weather Service (NWS) na monitoração de contexto do EXEHDA. In: ESCOLA REGIONAL DE ALTO DESEMPENHO (ERAD 2003), 2003, Santa Maria. **Anais....** Santa Maria: SBC/UFSM, 2003. p. 181-184 [COS 2003].

WSCAD 2003. Gustavo Frainer, Rodrigo Real, Adenauer Yamin, Luciano da Silva, Iara Augustin, Cláudio Geyer. Perfis Adaptativos para Balanceamento de Carga no ISAM. In: Workshop em Sistemas Computacionais de Alto Desempenho, 4, 2003, São Paulo, Brasil. **Proceedings…** São Paulo: USP, Novembro, 2003. p.164-167 [FRA 2003].

Publicações 2004

JCBS 2004. Adenauer Yamin, Iara Augustin, Luciano da Silva, Rodrigo Real, Jorge Barbosa, Cláudio Geyer. EXEHDA: an adaptive *middleware* for the pervasive computing scenery. **Journal of the Brazilian Computer Society**. Special Issue: Adaptive Software Systems. 2004. <submetido à publicação> [YAM 2004].

SOFTWARE, PRACTICE & EXPERIENCE. Special Issue on Experiences with Auto-adaptive and Reconfigurable Systems. Iara Augustin, Adenauer Yamin, Luciano C. Silva, Rodrigo Real, Claudio Geyer. ISAMadapt: abstractions and tools for designing general-purpose pervasive applications. Wiley InterScience. <submetido à publicação> [AUG 2004b].

SBLP 2004. Iara Augustin, Adenauer Yamin, Luciano C. Silva, Rodrigo Real, Claudio Geyer. ISAMadapt - um Ambiente de Desenvolvimento de Aplicações para a Computação Pervasiva. Simpósio Brasileiro de Linguagens de Programação (SBLP 2004), Niterói, Maio, 2004 [AUG 2004a].

IBERCHIP 2004. Lucas Brusamarello et al. Timing Verification Based on floating vector simulation: a distributed approach. In: X WORKSHOP IBERCHIP, 2004, Cartagena de Indias, Colombia. Março 2004 [BRU 2004].

CLEI 2004. Adenauer Yamin, Iara Augustin, Luciano C. Silva, Rodrigo Real, Jorge Barbosa, Cláudio Geyer. ISAM: Uma Arquitetura de Software para Pervasive Computing. Arequipa, Peru. Setembro 2004 (aceito para publicação) [YAM 2004a].

ERAD 2004. Gustavo Frainer, Rodrigo Real, Adenauer Yamin, Luciano da Silva, Iara Augustin, Cláudio Geyer. Perfis Adaptativos para Balanceamento de Carga no ISAM. In: ESCOLA REGIONAL DE ALTO DESEMPENHO (ERAD 2004), 2004, Pelotas. Anais... . Pelotas: SBC/UFPel/UCPel, 2004. p. 225-228 [FRA 2004].

8.2 Trabalhos futuros

O EXEHDA enquanto parte da arquitetura ISAM, que é abrangente, apresenta diversas oportunidades a serem exploradas. Algumas dessas oportunidades já foram traduzidas para trabalhos. Neste sentido, os principais temas de pesquisas em curso são:

- dotar o serviço de escalonamento do *middleware* de heurística que melhore a tomada de decisões em condições de incerteza. Esta abordagem é oportuna tendo em vista a dificuldade em se obter informações no tocante à efetiva disponibilidade dos recursos no âmbito da Computação Pervasiva. Este tópico é explorado na dissertação de mestrado em andamento de Real [REA 2004], a qual avalia o uso de redes bayesianas no âmbito do serviço *Scheduler*;
- aprofundar os estudos relativos à descoberta de recursos na Computação Pervasiva. Em particular, na dissertação de Schaeffer Filho [SCH 2004] o serviço *Discoverer* está sendo revisto na perspectiva da escalabilidade da estratégia de pesquisa, e da expressividade da linguagem empregada na descrição dos recursos;
- otimizar a estratégia de disseminação das informações provenientes da monitoração de recursos. Neste sentido, na dissertação de mestrado de Moraes [MOM 2004] o serviço *Deflector* está sendo revisto;
- reavaliar as estratégias de composição da informação de contexto e da gerência da adaptação, sob a óptica da agilidade do processo adaptativo. Tema este contemplado na proposta de doutorado de Silva [SIL 2004] direcionado aos serviços *ContextManager* e *AdaptEngine*;
- remodelar o serviço *ResourceBroker* para suporte à contabilização (*accounting*) considerando aspectos econômicos na alocação de recursos. Esta frente é contemplada na proposta de doutorado de Morais [MOL 2004].

Adicionalmente a estas atividades, vinculadas a programa de pós-graduação, outras frentes merecem atenção, as quais apresentam-se caracterizadas a seguir.

- aprofundar as pesquisas referentes ao serviço de reconhecimento de contexto do *middleware* – ISAMcontextService. Este trabalho está sendo desenvolvido no âmbito do projeto contextS, financiado pelo CNPQ/FINEP/SEPIN [CTX 2003];

- expandir a funcionalidade de módulos da ferramenta EXEHDA-AMI, em particular dos módulos para gerência de usuários e de configuração do serviço DC;
- desenvolver módulo para a EXEHDA-AMI, direcionado para visualização gráfica espacial dos recursos do ISAMpe, incorporando, além das informações sobre o estado dos recursos, uma representação da sua localização física;
- avaliar a interoperabilidade com outras ferramentas para gerência de arquiteturas distribuídas. Particularmente estudar a integração com a plataforma *Globus Metacomputing Tool Kit* [FOS 2001]. O Globus é atualmente uma das mais difundidas propostas para Computação em Grade;
- integrar o EXEHDA com ferramentas para depuração de execuções distribuídas. Nesta direção a ferramenta Pajé [STI 99], em desenvolvimento na UFSM, se mostra interessante, pois oportuniza integração com universidade da região;
- avaliar a especialização dos serviços do subsistema de comunicação para uso de APIs especializadas para plataformas de HPC (alto desempenho). Por exemplo, a API GM utilizada na rede Myrinet [MYR 2003];
- desenvolver *frameworks* direcionados a classes de aplicações mais usuais. Esta abordagem facilita o reuso de estratégias para distribuição e adaptação, entre as aplicações com características semelhantes;
- reavaliar os aspectos de segurança atualmente praticados, tanto na perspectiva da autenticação dos usuários, quanto na gerência dos recursos distribuídos utilizados e dos mecanismos de comunicação praticados.

O ISAM como um todo está em fase de expansão, com diversos tópicos de pesquisa sendo estudados. No que se refere ao EXEHDA, o estado atual do *middleware* contempla funcionalidades gerais para todos os serviços considerados básicos no suporte à Computação Pervasiva. O aprofundamento dos diversos serviços, via de regra, constitui oportunidades naturais de novos trabalhos de pesquisa. No que diz respeito à transposição dos esforços acadêmicos dedicados ao EXEHDA em resultados para a sociedade em geral, o projeto GRADEp: *Middleware* para gerenciar um ambiente de grade pervasiva (GT-RNP), constitui uma significativa oportunidade. Este projeto é centrado na disponibilização de soluções na forma de um produto, e está inserido em uma das principais infra-estruturas brasileiras de serviços e comunicações dedicada a pesquisa e desenvolvimento, que é a RNP.

REFERÊNCIAS

[AJI 2000] AJIB, W.; GODLEWSKI, P. A proposal of an Access persistence Protocol over Data Wireless Networks. In: IEEE INTERNATIONAL PERFORMANCE COMPUTING AND COMMUNICATIONS CONFERENCE, IPCCC 2000, Phoenix, Arizona. **Proceedings…** Disponível em: <http://www.infres.enst.fr/~wajib/publ.html>. Acesso em: maio 2001.

[ALB 2000] ALBURQUERQUE, Silvia Calmon de. **Gerenciamento do Consumo de Energia em Redes de Computação Móvel**, 2000. 98 f. Dissertação (Mestrado em Ciência da Computação) – Departamento de Ciência da Computação, UFMG, Belo Horizonte.

[ALI 2003] ALICE PROJECT. **Architecture for Location-Independent Computing Environments**. Disponível em: <http://www.dsg.cs.tcd.ie/research/alice/>. Acesso em: jan. 2003.

[ANG 98] ANGIN, O. et al. The Mobiware Toolkit: Programmable Support for Adaptive Mobile Networking. **IEEE Personal Communications Magazine** - Special Issue on Adapting to Network and Client Variability, New York, v.5, n.8, Aug. 1998.

[APE 2000] PROJETO APPELO. **Ambiente de Programação Paralela em Lógica**. Disponível em: <http://www.inf.ufrgs.br/procpar/opera/APPELO/>. Acesso em: nov. 2000.

[ARA 2001] ARAUJO, E.; AUGUSTIN, I.; YAMIN, A.; SILVA, L.; GEYER, C. Uma Proposta de Monitoração para Visualização de Aplicações Distribuídas Java, In: JORNADAS CHILENAS DE COMPUTACIÓN 2001, Puenta Arenas, Chile. **Anales...** Punta Arenas: Universidade de Magallanes, 2001. 1 CD-ROM.

[ARA 2002] ARAÚJO, E. **DOMonitor – um Ambiente de Monitoração de Aplicações Distribuídas Java.** 2002. 110f. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[AUG 2001] AUGUSTIN, I.; YAMIN, A.; BARBOSA, J.; GEYER, C. Requisitos para o Projeto de Aplicações Móveis Distribuídas. In: CONGRESO ARGENTINO DE CIENCIAS DE LA COMPUTACIÓN, CACIC, 8., 2001, El Calafate, Argentina. **Anales...** Buenos Aires: UNCPBA, 2001. 1 CD-ROM.

[AUG 2001a] AUGUSTIN, I.; YAMIN, A.; BARBOSA, J.; CAVALHEIRO, G.; GEYER, C. ISAM: um *Middleware* para Aplicações Móveis Distribuídas. **RITA – Revista de Informática Teórica e Aplicada**. Edição Especial – Sistemas Operacionais, Porto Alegre, v.8, n.2, p.41-58, fev. 2001.

[AUG 2002] AUGUSTIN, I.; YAMIN, A.; BARBOSA, J.; GEYER, C. Towards a Taxonomy for Mobile Applications with Adaptive Behavior. In: INTERNATIONAL SYMPOSIUM ON PARALLEL AND DISTRIBUTED COMPUTING AND NETWORKING, PDCN, 20., 2002, Innsbruck, Austria, **Proceedings…** Innsbruck: IASTED Press, 2002.

[AUG 2002a] AUGUSTIN, I.; YAMIN, A.; GEYER, C. Distributed Mobile Applications With Dynamic Adaptive Behavior. In: INTERNATIONAL CONFERENCE ON COMPUTERS AND THEIR APPLICATIONS, CATA, 17., 2002, San Francisco, Califórnia. **Proceedings…** Cary: ISCA, 2002.

[AUG 2002b] AUGUSTIN, I.; YAMIN, A.; BARBOSA, J.; GEYER, C.. ISAM - a Software Architecture for Adaptive and Distributed Mobile Applications. In: IEEE SYMPOSIUM ON COMPUTERS AND COMMUNICATIONS, ISCC, 7., 2002, Taormina, Italy. **Proceedings…** New York: IEEE Press, 2002.

[AUG 2003] AUGUSTIN, I.; YAMIN, A.; BARBOSA, J.; SILVA, L.; REAL, R.; GEYER, C. ISAM, Joing Context-awareness and Mobility to Building Pervasive Applications. In: MAHGOUB, I.; ILYAS, M. (Ed.). **Mobile Computing Handbook**. New York: CRC Press, 2003. p.73-94.

[AUG 2004] AUGUSTIN, I. **Abstrações para uma Linguagem de Programação visando Aplicações Móveis Conscientes do Contexto em um Ambiente de Pervasive Computing**. 2004. 193p. Tese (Doutorado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[AUG 2004a] AUGUSTIN, I.; YAMIN, A.; SILVA, L.; REAL, R.; GEYER, C.. ISAMadapt - um Ambiente de Desenvolvimento de Aplicações para a Computação Pervasiva. In: SIMPÓSIO BRASILEIRO DE LINGUAGENS DE PROGRAMAÇÃO, SBLP, 8., 2004, Niterói, **Anais ...** UFF: SBC, 2004. p.135-139.

[AUG 2004b] AUGUSTIN, I.; YAMIN, A.; SILVA, L.; REAL, R.; GEYER, C. ISAMadapt: abstractions and tools for designing general-purpose pervasive applications. Submetido a publicação a Software Practice & Experience: Special Issue on Experiences with Auto-adaptive and Reconfigurable Systems. Wiley InterScience. 2004.

[AZE 2001] AZEVEDO, S.; KAYSER, P.; YAMIN, A.; BARBOSA, J.; GEYER, C. DEPAnalyzer: um Analisador Estático de Dependências para Programas Java. In: WORKSHOP EM SISTEMAS COMPUTACIONAIS DE ALTO DESEMPENHO, WSCAD, 2., 2001. Pirenópolis **Anais…** . Pirenópolis: UNB:SBC, 2001. p. 135-141.

[BAG 98] BAGGIO, A. System Support for Transparency and Network-aware Adaptation in Mobile Environments In: ACM SYMPOSIUM ON APPLIED COMPUTING SPECIAL TRACK ON MOBILE COMPUTING SYSTEMS AND APPLICATIONS, 1., 1998, Atlanta. **Proceedings…** New York: ACM, 1998. p.405-408.

[BAL 89] BAL, H.; STEINER, J.; TANENBAUM, A. S. Programming Languages for Distributed Computing Systems. **ACM Computing Surveys**, New York, v.21, n.3, p.261-322, Sept. 1989.

[BAR 96] BARBOSA, J. GRANLOG: **Um Modelo para Análise Automática de Granulosidade na Programação em Lógica**. 1996. 167p. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[BAR 2000] BARBOSA, J. **Software Multiparadigma Paralelo e Distribuído.** 2000. 104 f. Exame de Qualificação (Doutorado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[BAR 2001] BARBOSA, J.; YAMIN, A.; VARGAS, P.; FERRARI, D.; SCHAEFFER, E.; GEYER, C. Using Mobility and Blackboards to Support a Multiparadigm Model Oriented to Distributed Processing. In: SYMPOSIUM ON COMPUTER ARCHITECTURE AND HIGH PERFORMANCE COMPUTING, SBAC-PAD, 13., 2001, Pirenópolis. **Proceedings…** Brasília: UNB, September 2001. p.187-194.

[BAR 2002] BARBOSA, J. **Holoparadigma:** Um Modelo Multiparadigma Orientado ao Desenvolvimento de Software Distribuído. 2002. 213 f. Tese (Doutorado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[BAR 2002a] BARBOSA, J.; YAMIN, A.; VARGAS, P.; AUGUSTIN, I.; GEYER, C. Holoparadigm: a Multiparadigm Model Oriented to Development of Distributed Systems In. INTERNATIONAL CONFERENCE ON PARALLEL AND DISTRIBUTED SYSTEMS, ICPADS, 9., 2002, Jung-Li City **Proceedings…** New York: IEEE Press, 2002. p. 165-171.

[BAY 97] BAYDERE, S. MaROS: a Framework for Application Development on Mobile Hosts. In: INTERNATIONAL CONFERENCE ON DISTRIBUTED AND PARALLEL SYSTEMS, EURO-PDS, 1997, Barcelona, Spain. **Proceedings…** Calgary, Canada: ACTA Press, 1997. p. 269-274.

[BEL 2001] BELLAVISTA, P.; CORRADI, A.; STEFANELLI, C. Mobile Agent *Middleware* for Mobile Computing. **IEEE Computer**, New York, v.34, n.3, p. 73-81, Mar. 2001.

[BEL 2003] BELLAVISTA, P.; CORRADI, A.; MONTANARI, R.; STEFANELLI, C. Dynamic Binding in Mobile Applications, a *Middleware* Approach. **IEEE Internet Computing**, Los Alamitos, v.7, n.2, p. 34-42, Mar-Apr, 2003.

[BHA 98] BHARGHAVAN, V. et al. The TIMELY Adaptive Resource Management Architecture. **IEEE Personal Communications Magazine**, New York, v.5, n.4, p. 20-31, aug. 1998.

[BIR 96] BIRMAN, K. P. **Building secure and reliable network applications**. Greenwich (US): Manning Publications Co., 1996.

[BLA 98] BLAIR, G. S. Na Architecture for Next Generation *Middleware*. In: ACM *MIDDLEWARE* CONFERENCE, 1998, Lake District, England. **Proceedings...** Berlin: Springer-Verlag, 1998.

[BLU 2002] BLUETOOTH Website. How it Works. Disponível em: <http://www.bluetooth.com/>. Acesso em: mar. 2002.

[BRI 90] BRIAT, J.; FAVRE, M.; GEYER, C. et al. **Opera:** a Parallel Prolog System and its Implementation on Supernode. 1990. 87 f. (Technical Report) - Laboratoire de Genie Informatique de Grenoble/CAP-Gemini-Innovation, Grenoble.

[BOR 2002] BORLAND INTERNATIONAL. **Visibroker Enterprise Server**. Disponível em: <http://www.borland.com/besvisibroker/index.htm>. Acesso em: mar. 2002.

[BRI 90a] BRIAT, J.; FAVRE, M.; GEYER, C. et al. **Scheduling of OR-parallel Prolog on Scalable, Reconfigurable, Distributed-Memory Multiprocessor**. 1990. 67 f. (Technical Report) - Laboratoire de Genie Informatique de Grenoble/CAP-Gemini-innovation, Grenoble.

[BRU 2003] BRUSAMARELLO, L. et al. Timing Verification Based on Floating Vector Simulation on a Distributed System. In: SOUTH SYMPOSIUM ON MICROELETRONICS, SIM, 18., 2003, Novo Hamburgo. **Proceedings…** Novo Hamburgo: Ed. Feevale, 2003.

[BRU 2004] BRUSAMARELLO, Lucas et al. Timing Verification Based on floating vector simulation: a distributed approach. In: WORKSHOP IBERCHIP, 10., 2004, Cartagena de Índias. **Resumos...** Cartagena de Índias: INALDE, 2004. 1 CD-ROM.

[BRU 2004a] BRUSAMARELLO, L. et al. Um Sistema Distribuído para Posicionamento de Células em Circuitos VLSI. In: ESCOLA REGIONAL DE ALTO DESEMPENHO, ERAD, 2004, Pelotas. **Anais...**. Pelotas: SBC/UCPel/UFPel, 2004. p. 237-240.

[BUN 2000] BUNDSCHUH, R. An analytic approach to significance assessment in local sequence alignment with gaps. In: INTERNATIONAL CONFERENCE ON COMPUTATIONAL MOLECULAR BIOLOGY, RECOMB, 4., 2000, Tokyo. **Proceedings…** New York:ACM Press, 2000. p. 86-95.

[BUS 96] BUSCHMANN, Frank. **Pattern-oriented software arquitecture :** a system of patterns. New York: John Wiley. 1996.

[BUY 2001] BUYYA, R. et al. **Architectural Models for Resource Management in the Grid**. Disponível em: <http://citeseer.nj.nec.com /321320.html>. Acesso em: out. 2001.

[CAB 2002] CABRI, G.; LEONARDI, L.; ZAMBONELLI, F. Engineering Mobile Agent Applications via Context-Dependent Coordination. **IEEE Transactions on Software Engineering**, Los Alamitos, v.28, n.11, nov. 2002. p. 1039-1055.

[CAR 2002] CARDOZO JR, Marcelo Augusto Cardozo. EXEHDA-CC: **Uma Contribuição à Comunicação e a Coordenação no EXEHDA**. 2002. 61 f. Projeto de Diplomação (Bacharelado em Ciência da Computação) – Curso de Informática, UFPel, Pelotas.

[CAS 97] CASTRO, Luis .Fernando.Pias de. **Um Modelo de Analisador Estático Baseado na Interpretação Abstrata Direcionado à Paralelização de Programas em Lógica**. 1997. 69p. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[CAS 98] CASANOVA, H. DONGARRA J. NetSolve: Network enabled solvers, **IEEE Computational Science and Engineering**, Los Alamitos, v.5, n.3, p. 57-67, Sept. 1998.

[CEN 97] CEN, S. **A Software Feedback Toolkit and its Application in Adaptive Multimedia Systems**. 1997. 213p. Ph.D. Thesis - Department of Computer Science and Engineering, Oregon Graduate Institute of Science and Technology. Disponível em: <http://www.cse.ogi.edu/~scen/Shanwei-Cen-PhD-Thesis.ps.gz>. Acesso em: out. 2001.

[CEN 99] CENTENO, Ana Paula. **Penelope - Um Modelo de Escalonador Hierárquico para Sistemas que Exploram o Paralelismo OU em Programas em Lógica**. 1999. 65p. Dissertação (Mestrado.em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[CER 2003] CERCATO, F.; SANTIN, M. **EXEHDA-WP:** Primitivas para suporte ao EXEHDA no ambiente Windows. 2003. 67 f. Projeto de Diplomação (Bacharelado em Ciência da Computação) – Escola de Informática, UCPel, Pelotas.

[CHE 2001] CHEN, Guanling; KOTZ, David. **A Survey of Context-aware Mobile Computing Research.** (Technical Report TR2000-381). Dartmouth Computer Science. Disponível em: <http://www.cs.dartmouth.edu/~solar/>. Acesso em: mar. 2001.

[CIC 2002] CICS. Customer Information Control System. Disponível em: <http://www-3.ibm.com/software/htp/cics/>. Acesso em: ago. 2002.

[CLA 2001] CLARKE, M. et al. An Efficient Component Model for the Construction of Adaptive *Middleware*. In: IFIP/ACM INTERNATIONAL CONFERENCE ON DISTRIBUTED SYSTEMS PLATFORMS, 2001, Heidelberg. **Proceedings …** London:Springer-Verlag, 2001. p. 160-178.

[COR 2002] CORBA. Common Object Request Broker Architecture - Catalog of OMG CORBA/IIOP Specifications. Disponível em: <http://www.omg.org/technology/documents/corba_speccatalog.htm>. Acesso em: set. 2002.

[COS 2003] COSTA JUNIOR, F. da; YAMIN, A.; BARBOSA, J.; GEYER, C.; AUGUSTIN, I. Uso do Network Weather Service (NWS) na monitoração de contexto do EXEHDA. In: ESCOLA REGIONAL DE ALTO DESEMPENHO, ERAD, 3., 2003, Santa Maria. **Anais...**. Santa Maria: SBC:UFSM, 2003. p. 181-184.

[COS 97] COSTA, C. **Um Modelo de Escalonamento Hierárquico para Prolog Paralelo**. 1997. 104 f. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[COS 2003a] COSTA, F. **Desenvolvendo Aplicações Sensíveis ao Contexto na *Pervasive Computing***. 2003. 50 f. Projeto de Diplomação (Bacharelado em Ciência da Computação) – Escola de Informática, UCPel, Pelotas.

[COU 2001] COULOURIS, George; DOLLIMORE, Jean; KINDBERG, Tim. **Distributed Systems:** Concepts and Design. Harlow: Addison-Wesley, 2001. 772p.

[CTX 2003] CONTEXTS**.** Um *Middleware* para Desenvolvimento de Aplicações Sensíveis ao Contexto. Projeto aprovado na chamada conjunta do SEPIN/MCT - FINEP - CNPq 01/2002. Programa PDI-TI. Fundo Setorial de Informática. Em desenvolvimento na UFRGS, UFSM e UCPel. Financiamento em Janeiro de 2003.

[CUG 2001] CUGOLA, G.; NITTO, E. Using a publish/subscribe middleware to support mobile computing. In: ACM MIDDLEWARE 2001 CONFERENCE, 2001, Germany. **Proceedings…** Berlin:Springer-Verlag, 2001.

[CUG 2001a] CUGOLA, G.; PICCO, G.P. **Peerware:** Core *Middleware* Support for Peer-to-Peer and Mobile Systems. Technical Report, Politecnico di Milano, Italy, 2001. Disponível em: < www.elet.polimi.it>. Acesso em: dez. 2002.

[DAV 97] DAVIES, N. et al. Limbo: a Tuple Space Based Platform for Adaptive Mobile Applications. In: INTERNATIONAL CONFERENCE ON OPEN DISTRIBUTED PROCESSING/DISTRIBUTED PLATFORMS, ICODP/ICDP, 1997, Toronto. **Proceedings…** [S.l.:s.n.], 1997.

[DEY 2000] DEY, A. K.; ABOWD, G. D. Towards a Better Understanding of Context and Context-Awareness. In: CONFERENTE ON HUMAN FACTORS IN COMPUTING SYSTEMS, HCI2000, 14, 2000. UK. **Proceedings…** New York:ACM Press, 2000.

[DEY 2001] DEY, A. K. Understanding and Using Context. **Personal and Ubiquitous Computing**, New York, v.5, n.1, p. 4-7, 2001.

[DNS 2002] DNS RESOURCES DIRECTORY. Disponível em: <http://www.dns. net/dnsrd/>. Disponível em: <http://dsonline.computer.org>. Acesso em: fev. 2002.

[DOL 2003] DOLMEN Project. Disponível em: <www.fub.it/dolmen>. Acesso em: mar. 2003.

[ELI 2002] ELIASSEN, F. et al. **QoS management in the MULTE-ORB. IEEE Distributed Systems Online**. Disponível em: <http://dsonline.computer.org>. Acesso em: jul. 2002.

[EMM 2000] EMMERICH, W. Software Engineering and *Middleware*: a Roadmap. The Future of Software Engineering. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, ICSE, 22., Limerick, Ireland, 2000. **Proceedings…** Berlin:Springer-Verlag, 2000.

[EXE 2002] EXEHDA Unicluster. A Internet Platform for Distributed Computing. **Open Source Project: sources, binaries and documentation.** Disponível em: <http://savannah.nongnu.org/projects/unicluster>. Acesso em: mar. 2002.

[EXE 2003] EXEHDA. Execution Environment for Highly Distributed Applications. Disponível em: <http://www.inf.ufrgs.br/ ~exehda/>. Acesso em: ago. 2003.

[FAR 98] FARLEY, J.; LOUKIDES, M. **Java Distributed Computing**. London: O'Reilly, 1998.

[FEH 2003] FEHLBERG, F. **ISAM e seu uso em simulações numericamente intensivas**. 2003. 59 f. Projeto de Diplomação (Bacharelado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[FER 2001] FERRARI, D. **Um Modelo de Replicação em Ambientes que Suportam Mobilidade**. 2001. 88p. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[FOS 98] FOSTER, I.; KESSELMAN, C. The Globus Project: A Status Report. In: INTERNATIONAL PARALLEL PROCESSING SYMPOSIUM, IPPS/SPDP, 12., 1998, Orlando. **Proceedings…** New York: IEEE Press, 1998.

[FOS 99] FOSTER, I.; KESSELMAN; C, (Ed.) **The Grid**: Blueprint for a New Computing Infrastructure. San Francisco: Morgan Kaufmann Publishers, 1999.

[FOS 2001] FOSTER, I.; KESSELMAN, C.; TUECKE, S. The anatomy of the Grid: enabling scalabe virtual organization. **International Journal of High Performance Computing Applications**, New York, v.15, n.3, p.200-222, May 2001.

[FOX 98] FOX, A. et al. Adapting to Network and Client Variation Using Infrastructural Proxies: Lessons and Perspectives. **Personal Communications**, New York, v.5, n.4, p. 10-19, Aug. 1998.

[FRA 2003] FRAINER, G.; REAL, R.; YAMIN, A.; SILVA, L.; AUGUSTIN, I.; GEYER, C. Perfis Adaptativos para Balanceamento de Carga no ISAM. In: WORKSHOP EM SISTEMAS COMPUTACIONAIS DE ALTO DESEMPENHO, 4., 2003, São Paulo. **Anais…** São Paulo: USP, 2003. p.164-167.

[FRA 2004] FRAINER, G.; REAL, R.; YAMIN, A.; SILVA, L.; AUGUSTIN, I.; GEYER, C. Perfis Adaptativos para Balanceamento de Carga no ISAM. In: ESCOLA REGIONAL DE ALTO DESEMPENHO, ERAD, 4., 2004, Pelotas. **Anais...** . Pelotas: SBC:UFPel:UCPel, 2004. p. 225-228.

[FRI 2000] FRITSCH, D.; KLINEC, D.; VOLZ, S. NEXUS Positioning and Data Management Concepts for Location Aware Applications. In: INTERNATIONAL SYMPOSIUM ON TELEGEOPROCESSING, 2., 2000, Nice, France. **Proceedings**…[S.l.:s.n.], 2000.

[FRZ 2003] FRANZ, D.; CARDOZO JUNIOR, M.; BARBOSA, J.; YAMIN, A.; GEYER, C. EXEHDA-CC x JADA: Uma análise de espaço de objetos compartilhados em Java. In: ESCOLA REGIONAL DE ALTO DESEMPENHO, ERAD, 3., 2003, Santa Maria. **Anais...** . Santa Maria: SBC:UFSM, 2003. p. 177-180.

[FUG 98] FUGETTA, A.; PICCO, G. P.; VIGNA, G. Understanding Code Mobility. **IEEE Transactions on Software Engineering**, New York, v.24, n.5, p. 342-36**,** May 1998.

[FUH 2002] FUHRMANN, T. et al. AMnet 2.0: An Improved Architecture for Programmable Networks. In: INTERNATIONAL WORKING CONFERENCE ON ACTIVE NETWORKS, IWAN2002, 4., 2002. Zurich. **Proceedings…** London:Springer-Verlag, 2002. p. 162-176.

[GAM 97] GAMMA, E. et al. **Design Patterns:** elements of reusable object-oriented software**.** 10th ed. Reading, MA: Addison-Wesley, 1997.

[GAR 2002] GARLAN, D.; STEENKISTE, P.; SCHMERL, B. Project Aura: Toward Distraction-free *Pervasive Computing*. **IEEE *Pervasive Computing*,** New York, v.1, n.2, p. 22-31, Apr. 2002.

[GEA 2003] PALMGEAR. **Software for PDAs**. Disponível em: <http://www.palmgear.com/>. Acesso em: jan. 2003.

[GEL 85] GELERNTER, D.; CARRIERO, N. Generative Communication in Linda. **ACM Transactions on Programming Languages and Systems**, New York. v.7, n.1, p.80-112, Jan. 1985.

[GET 2002] GETOV, V. et al. **Multi-Paradigm Communications in Java for Grid Computing**. Disponível em: <http://www.javagrande.org/ jgpapers.html>. Acesso em: mar. 2002.

[GEY 92] GEYER, C.; YAMIN, A.; WERNER, O. Projeto OPERA: Um Modelo E/OU para Prolog. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 12., 1992. Florianópolis. **Proceedings…** Porto Alegre:SBC, 1992. p.269-281.

[GEY 99] GEYER, C.; YAMIN, A.; BARBOSA, J.; COSTA, C. et al. The APPELO Project - Parallel Environment for Logic Programming. In: PROJECTS EVALUATION WORKSHOP FASE III, PROTEM-CC, 1999, Rio de Janeiro. **Proceedings…** Rio de Janeiro: CNPQ, 1999. p. 421-454.

[GEY 2002] GEYER, C.; YAMIN, A.; SILVA, L. da; VARGAS, P.; DUTRA, I.; PETEK, M.; ADAMATTI, D.; AUGUSTIN, I.; BARBOSA, J. Regional Center And Grid Development In Brazil. In: LAFEX INTERNATIONAL SCHOOL ON HICH ENERGY PHYSICS, LISHEP2002**,** 2002, Rio de Janeiro. **Proceeding…** [S.l.:s.n.], 2002. Disponível em: <http://www.inf.ufrgs.br/~exehda/>. Acesso em: dez. 2002.

[GON 2004] GONÇALVES, A.; VIGNOLI, F. **Segurança de Redes na Pervasive Computing**. 2004. 128 f. Projeto de Diplomação (Bacharelado em Ciência da Computação) – Escola de Informática, UCPEL, Pelotas.

[GPB 2003] GRUPO de Usuários de Palm-BR. Disponível em: <http://www.palm-br.com.br/>. Acesso em: jan. 2003.

[GRI 2001] GRIMM, Robert et al. Programming for *Pervasive Computing* Environment. In: SYMPOSIUM ON OPERATING SYSTEM PRINCIPLES, 18., 2001. Canada. **Proceedings...** Canada: ACM, 2001.

[GRI 2001a] GRIMM, R. et al. System Direction for *Pervasive Computing*. In: IEEE WORKSHOP ON HOT TOPICS IN OPERATING SYSTEMS, 8., 2001, Elmau. **Proceedings…** [S.l.:s.n.], 2001. Disponível em: <http://www.cs.nyu.edu/rgrimm/publications.html>. Acesso em: mar. 2001.

[GRI 97] GRIMSHAW, A; et al. The Legion Vision of a World-Wide Virtual Computer. **Communications of the ACM**, New York, v.40, n.1, p. 39-45, Jan. 1997.

[GRO 99] GROSS, T.; STEENKISTE, P.; SUBHLOK, J. Adaptive Distributed Applications on Heterogeneous Networks. In: HETEROGENEOUS COMPUTING WORKSHOP, HCW, 8., San Juan, Puerto Rico, 1999. **Proceedings...** New York:IEEE Computer Society, 1999.

[GSM 2002] USING GSM. Disponível em: <http://www.gsmworld.com /index.shtml>. Acesso em: out. 2002.

[HEN 2001] HENRICKSEN, K.; INDULSKA, J.; RAKOTONIRAINY, A. Infrastructure for Pervasive Computing: Challenges. In: WORKSHOP ON PERVASIVE COMPUTING, INFORMATIK, 1., 2001, Vienna. **Proceedings…** [S.l.:s.n.], 2002. Disponível em: <http://www.dstc.edu.au/m3/papers/Informatik2001.pdf>. Acesso em: nov. 2001.

[HEN 2002] HENRICKSEN, K.; INDULSKA, J.; RAKOTONIRAINY, A. Modeling Context Information in Pervasive Computing System. In: PERVASIVE 2002, Berlin, Germany. **Proceedings…** Berlin:Springer-Verlag, 2002. (Lecture Notes in Computer Science, 2414).

[HES 2002] HESS, C. K.; ROMAN, M.; CAMPBELL, R. Building Applications for Ubiquitous Computing Environments. In: CONFERENCE ON PERVASIVE COMPUTING, 2002, Zurich, Switzerland. **Proceedings**…[S.l.:s.n.], 2002.

[HOL 2000] HOLOPARADIGMA. **Software Multiparadigma Distribuído**. Disponível em: <http://www.inf.ufrgs.br/~holo/>. Acesso em: nov. 2002.

[HOU 2002] HOUSLEY, R. et al. **Internet X.509 Public Key Infrastructure Certificate and CRL Profile.** Disponível em: <http://www.ietf.org/ rfc/rfc2459.txt>. Acesso em: abr. 2002.

[IBM 99] PERVASIVE Computing. **IBM System Journal**, New York, v.38, n.4, 1999. Disponível em: <http://www.research.ibm.com/ journal/sj38-4.html>. Acesso em: set. 1999.

[GRI 95] GRISWOLD, W.; NOTKIN, D. **IEEE Transactions on Software Engineering**, New York, v.21, n.4, p. 275-287, Apr. 1995.

[IET 2001] IETF RFC2768 - Request for Comments 2768. ***Middlewares***. Disponível em: <http://www.globecom.net/ietf/rfc/rfc2768.html>. Acesso em: ago. 2001.

[IET 2002] IETF. **The Internet Engineering Task Force**. Disponível em: <http://www.ietf.org/>. Acesso em: jul. 2002.

[ION 2002] IONA TECHNOLOGIES. **Orbix product family**. Disponível em: <http://www.corba.org/vendors/ pages/iona.html>. Acesso em: jun. 2002.

[IPC 2002] IEEE Pervasive Computing. **Mobile and Ubiquitous Systems**. Disponível em: <http://www.computer.org/pervasive/>. Acesso em: mar. 2002.

[ISA 2002] ISAM. Infra-Estrutura de Suporte às Aplicações Móveis. Disponível em: <http://www.inf.ufrgs.br/~isam/>. Acesso em: ago. 2003.

[JME 2002] JINIME. Jini Mobile Edition - Research Proposal. Disponível em: <http://www.cs.rit.edu/~anhinga/whitepapers/ JiniMEProposal/>. Acesso em: dez. 2002.

[JMS 2002] JAVA TECNOLOGIES. **Java Message Service**. Disponível em: <http://java.sun.com/products/jms/>. Acesso em: abr. 2002.

[JOS 97] JOSEPH, A. D.; TAUBER, J. A.; KAASHOEK, M. F. Mobile Computing with Rover Toolkit. **IEEE Transactions on Computers**, New York, v.43, n.3, p. 337-352, Mar. 1997.

[KAT 94] KATZ, R. H. Adaptation and Mobility in Wireless Information Systems. **IEEE Personal Communications**, New York, v.1, n.1, p.6-17, Jan. 1994.

[KNO 2003] KNOPPIX. Live GNU/Linux System. Disponível em: <http://www.knoppix.org/>. Acesso em: jul. 2003.

[KON 2000] KON, Fábio et al. Monitoring, Security, and Dynamic Configuration with the dynamicTAO Reflective ORB. In: IFIP/ACM INTERNATIONAL CONFERENCE ON DISTRIBUTED SYSTEMS PLATFORMS AND OPEN DISTRIBUTED PROCESSING, 2000, New York. **Proceedings …** New York:Springer-Verlag, 2000.

[KOR 2001] KORKEA-AHO, M. **Context-Aware Applications Survey**. Helsinki University of Technology. Disponível em: <http://www.hut.fi/ ~mkorkeaa /doc/context-aware.htm>. Acesso em: set. 2001.

[KRA 2001] KRAUTER, K. et al. **A Taxonomy and Survey of Grid Resource Management Systems for Distributed Computing**. Disponível em: <http://citeseer.nj.nec.com/krauter01taxonomy.html>. Acesso em: set. 2001.

[KUN 99] KUNZ, T.; BALCK, J. P. An Architecture for Adaptive Mobile Applications. In: INTERNATIONAL CONFERENCE ON PERSONAL WIRELESS COMMUNICATIONS, 11., 1999, Calgary, Alberta, Canada. **Proceedings…** New York:IEEE Computer Society, 1999.

[LED 99] LEDOUX, T. OpenCorba: A Reflective Open Broker. In: INTERNATIONAL CONFERENCE ON META-LEVEL ARCHITECTURES AND REFLECTION, REFLECTION99, 2., 1999 Saint-Malo, France. **Proceedings…** London: Springer-Verlag, 1999. p. 197 – 214.

[LIN 97] LINDHOLM, T.; YELLIN, F. **The Java Virtual Machine Specification**. Reading: Addison-Wesley, 1997.

[LIT 2001] LITIU, R.; PRAKASH, A. DACIA: A Mobile Component Framework for Building Adaptive Distributed Applications. In: IFIP/ACM INTERNATIONAL CONFERENCE ON DISTRIBUTED SYSTEMS PLATFORMS AND OPEN DISTRIBUTED PROCESSING, 2000, New York. **Proceedings …** New York:Springer-Verlag, 2000.

[LOW 98] LOWEKAMP, B. et al. A Resource Query Interface for Network-aware Applications. IEEE SYMPOSIUM ON HIGH-PERFORMANCE DISTRIBUTED COMPUTING, 7., 1998. Chicago. **Proceedings…** New York:IEEE Computer Society, 1998.

[LUN 2001] LUNARDI, S. **Uma Camada de Suporte à Qualidade de Serviço para Aplicações Multimídia na Internet.** 2001. 99p. Dissertação (Mestrado em Ciência da Computação) - Faculdade de Informática, PUCRS, Porto Alegre.

[MAM 2003] MAMEI, M.; ZAMBONELLI, F.; LEONARDI, L. Programming Ubiquitous and Mobile Computing Applications with TOTA. **IEEE Distributed System Online**, v.4, n.5, May 2003. Disponível em: <http://csdl.computer.org/comp/mags/ds/2003/ 05/o5toc.htm>. Acesso em: out. 2003.

[MAN 2003] MANETs. **Publications on Mobile Ad-hoc Networks**. Disponível em: <http://www.ee.surrey.ac.uk/Personal/G.Aggelou/MANET_ PUBLICATIONS.html> . Acesso em: set. 2003.

[MAR 2004] MARTINS, M. **Empregando Estratégias Context-Aware em jogos na Pervasive Computing:** um estudo de caso sobre o FreeMMG. 2004. 34 f. Projeto de Diplomação (Bacharelado em Ciência da Computação) – Escola de Informática, UCPel, Pelotas.

[MEI 94] MEIDANIS, J.; SETUBAL, J. Uma Introdução à Biologia Computacional. In: ESCOLA DE COMPUTAÇÃO, 9., 1994. Recife. **Anais ...** Recife:SBC, 1994.

[MEN 97] MENEZES, A. et al. **Handbook of applied cryptography**. Boca Raton, Florida: CRC Press, 1997.

[MIL 2000] MILLER N.; STEENKISTE, P. Collecting Network Status Information for Network-aware Applications. In: INFOCOM, 2000, Tel Aviv. **Proceedings…**[S.l.:s.n.], 2000.

[MOL 2004] MORAIS, L. **Accouting-ware:** um serviço para accouting e negociação de recursos direcionado à Computação Pervasiva. 2004. 67 f. Monografia (Proposta de Doutoramento) – Instituto de Informática, UFRGS, Porto Alegre.

[MOM 2004] MORAES, M. **Técnicas de disseminação de informações de monitoramento em sistemas largamente distribuídos**. 2004. Dissertação (Mestrado em Ciência da Computação em Andamento) – Instituto de Informática, UFRGS, Porto Alegre.

[MOR 99] MORRISON, M. **Como Programar em JavaBeans**. São Paulo: Makron Books do Brasil, 1999.

[MQS 2002] MQSeries. **WebSphere MQ family**. Disponível em: <http://www-3.ibm.com/software/ts/mqseries/>. Acesso em: set. 2002.

[MQS 2002a] MQSeries. **Information Road Map**. Disponível em: <http://www.developer.ibm.com/en_US/tech/faq/aux/mqroad.html>. Acesso em: abr. 2002.

[MSC 2002] MICROSOFT COM. **MS Component Object Model**. Disponível em: <http://www.microsoft.com/com/>. Acesso em: set. 2002.

[MYR 2003] MYRICOM INC. M. I. **GM Reference Manual**. 2003. White Paper. Disponível em: <http://www.myri.com/scs/GM/doc/refman.pdf>. Acesso em: out. 2003.

[NOB 2000] NOBLE, B. System Support for Mobile, Adaptive Applications. **IEEE Personal Computing Systems**, New York, v.7, n.1, p. 44-9, Feb. 2000.

[NOG 2001] NOGUEIRA, M.; YAMIN, A.; VARGAS, P.; GEYER, C. Balanceamento de Carga em Sistemas Distribuídos: Uma Proposta de Ambiente para Avaliação. In: CONFERÊNCIA LATINOAMERICANA DE INFORMÁTICA, 27., 2001, Mérida, Venezuela. **Proceedings…** Mérida: Universidad de Los Andes, 2001.

[OLD 2003] OPENLDAP Project. Lightweight Directory Access Protocol. Disponível em: <http://www.openldap.org/>. Acesso em: jul. 2003.

[OMG 2002] OBJECT MANAGEMENT GROUP. **CORBA Component Model**. Disponível em: <http://www.corba.org/>. Acesso em: mar. 2002.

[OPE 99] Projeto OPERA. **Prolog Paralelo**. Disponível em: <http://www.inf.ufrgs.br/ procpar/opera/OPERA/index.html/>. Acesso em: nov. 1999.

[PER 2003] PERES, R. **Estratégias Context-Aware para o Aumento de Desempenho em Servidores Web**. 2003. 59 f. Projeto de Diplomação (Bacharelado em Ciência da Computação) – Escola de Informática, UCPEL, Pelotas.

[PHA 98] PHAM, V. A.; KARMOUCH, A. Mobile Software Agents: an Overview. **IEEE Communications Magazine**, New York, v.36, n.7, p. 26-37, July 1998.

[PIC 2001] PICCO, G. P.; MURPHY, A. L.; ROMAN, G-C. Lime: a Middleware for Physical and Logical Mobility. In: INTERNATIONAL CONFERENCE ON DISTRIBUTED COMPUTING SYSTEMS, 2001, Phoenix, USA. **Proceedings…** Berlin:Springer-Verlag, 2001.

[RAK 2002] RAKOTONIRAINY, A.; GROVES, G. Resource Discovery for Pervasive Environment. In: INTERNATIONAL SYMPOSIUM ON DISTRIBUTED OBJECTS AND APPLICATION, DOA2002, 4., 2002, Irvine, USA. **Proceedings…** New York:IEEE Computer Society, 2002.

[RAN 97] RANGANATHAN, M.; ACHARYA, A.; SALTZ, J. Sumatra: a Language for Resource-aware Mobile Programs. **Mobile Objects Systems:** Towards the Programmable Internet. Berlin:Springer-Verlag. 1997. p. 111-130. (Lecture Notes in Computer Science, 1222).

[REA 2001] REAL, R.; DUARTE, N.; YAMIN, A. **UniCluster-EXEHDA: Uma Proposta de ICP Voltada para PAD**. 2001. 82 f. Projeto de Diplomação (Bacharelado em Engenharia da Computação) – Curso de Eng. da Computação, FURG, Rio Grande.

[REA 2002] REAL, R.; REAL, M.; DUARTE, N.; SALVADOR, O.; YAMIN, A. Unicluster: uma Proposta de ICP para PAD. In: ESCOLA REGIONAL DE ALTO DESEMPENHO, 2., 2002, São Leopoldo. **Anais...** Porto Alegre: SBC:UFRGS:UNISINOS:ULBRA, 2002. p.295-298.

[REA 2002a] REAL, R. **Avaliando a Utilização do *Globus Metacomputing Toolkit* no Projeto EXEHDA**. 2002. 69 f. Trabalho Individual (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[REA 2003] REAL, R.; YAMIN, A.; AUGUSTIN, I.; BARBOSA, J.; SILVA, L.; FRAINER, G.; GEYER, Cl. Tratamento da incerteza no escalonamento de recursos em *Pervasive Computing*. In: CONFERÊNCIA IADIS IBERO-AMERICANA WWW/INTERNET, 2003, Algarve, Portugal. **Anais...** Algarve: IADIS Press, 2003. p.167-170.

[REA 2003a] REAL, R.; YAMIN, A.; AUGUSTIN, I.; BARBOSA, J.; SILVA, L.; FRAINER, G.; GEYER, C. Resource scheduling on grid: handling uncertainty. In: IEEE/ACM INTERNATIONAL WORKSHOP ON GRID COMPUTING, 4., 2003, Phoenix, Arizona. **Proceedings…** New York: IEEE Press, 2003. p. 205-208.

[REA 2004] REAL, R. **Uma proposta de escalonamento para Computação Pervasiva**. 2004. 84 f. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[REI 2002] REIS, R.; REIS, C.; AUGUSTIN, I.; YAMIN, A.; GEYER, C.; NUNES, D. Towards a Software Process Model to Support the Design of Mobile Computing Applications. In: WORLD CONFERENCE ON INTEGRATED DESIGN AND PROCESS TECHNOLOGY, IDPT, 6., 2002. Pasadena, California, USA. **Proceedings…** [S.l.: s.n.], 2002.

[RIM 2002] RIMASSA, G. Wired-Wireless Integration: A *Middleware* Perspective. **IEEE Distributed System OnLine**. Disponível em: <http://dsonline.computer.org/0209/d/w5icon.htm>. Acesso em: set. 2002.

[RIN 98] RINE, D. C. Supporting Reuse With Object Technology. **IEEE Computer**, New York, v.30, n.10, p.43-45, Oct. 1998.

[RNP 2004] REDE NACIONAL DE ENSINO E PESQUISA – RNP. Promovendo o Uso Inovador de Redes Avançadas no Brasil. Disponível em: <http://www.rnp.br/>. Acesso em: abr. 2004.

[ROM 2000] ROMAN, G-C.; PICCO, G. P.; MURPHY, A. L. A Software Engineering Perspective on Mobility. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 22., 2000, Limerick, Ireland. **Proceedings…** New York:ACM Press, 2000. p. 241-258.

[ROM 2002] ROMAN, M. et al. Gaia: a *Middleware* Infrastructure to Enable Active Spaces. **IEEE *Pervasive Computing***, New York, v.1, n. 4, p.74-83, Oct. 2002.

[ROM 2003] ROMAN, M.; A**n Application Framework for Active Space Applications.** 2003. 186 p. Thesis (Doctoral Programm) - University of Illinois, Urbana-Champaign, Illinois, USA.

[SAH 2003] SAHA, D.; MUKHERJEE, A. *Pervasive Computing*: a Paradigm for th 21st Century**. IEEE Computer**, New York, v.36, n.3, p.25-31, Mar. 2003.

[SAT 2001] SATYANARAYANAN, M. *Pervasive Computing*: Vision and Challenges. **IEEE Personal Communications**, New York, v.8, n.4, p.10-17, 2001.

[SAT 90] SATYANARAYANAN, M; et al. Coda: a Highly Available File System for a Distributed Workstation Environment. **IEEE Transactions on Computers**, New York, v.39, n.4, p. 447-459, Apr. 1990.

[SAT 96] SATYANARAYANAN, M. Mobile Information Access. **IEEE Personal Communications**, New York, v.3, n.1, p.26-33, Feb. 1996.

[SCH 2002] SCHAFFER FILHO, Alberto Egon. **EXEHDA-HM**: Uma Contribuição à Gerência da HoloTree**.** 2002. 65 f. Projeto de Diplomação (Bacharelado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[SCH 2004] SCHAFFER FILHO, A. **Descoberta de Recursos no ISAM Pervasive Environment.** 2004 Dissertação (Mestrado em Ciência da Computação em andamento) – Instituto de Informática, UFRGS, Porto Alegre.

[SCH 2004a] SCHAFFER FILHO, A.; MORAIS, L.; REAL, R.; SILVA, L.; YAMIN, A.; AUGUSTIN, I.; GEYER, C. A Practical Grid Experience Using the ISAM Architecture for Genetic Sequence Alignment. In: ACM/IFIP/USENIX INTERNATIONAL MIDDLEWARE CONFERENCE, Middleware, 2004, Toronto, Canada. **Proceeding…** Berlin:Springer-Verlag, 2004. (Lecture Notes in Computer Science, 3231).

[SEA 97] SEAN, B. **Corba distributed objects:** using orbix. New York: ACM Press, 1997.

[SHA 96] SHAW, M.; GARLAN, D. **Software Architecture:** Perspectives on an Emerging Discipline. New Jersey: Prentice-Hall, 1996.

[SHA 2000] SHAO G.; BERMAN F.; and WOLSKI, R. **Performance Effects of Scheduling Strategies for Master/Slave Distributed Applications**, (UCSD Technical Report #CS98-598) Disponível em: <http://apples.ucsd. edu/pubs/cs98-598.ps>. Acesso em: oct. 2000.

[SIL 2001] SILVA, L. da.; YAMIN, A.; AUGUSTIN, I.; BARBOSA, J.; GEYER, C. Mecanismos de Suporte ao Escalonamento em Sistemas com Objetos Distribuídos Java. CACIC CONGRESO ARGENTINO DE CIENCIAS DE LA COMPUTACIÓN, 8., 2001, Santa Cruz, Argentina. **Proceedings…** [S.l.:s.n.], 2001. 1 CD-ROM.

[SIL 2003] SILVA, L. **Primitivas para Suporte à Distribuição de Objetos Direcionados à *Pervasive Computing***. 2003. 67 f. Dissertação (Mestrado em Ciência da Computação) - Instituto de Informática, UFRGS, Porto Alegre.

[SIL 2004] SILVA, L. **Adaptação na Computação Pervasiva**: modelo para tratamento da sensibilidade ao contexto enfocando o aspecto agilidade da adaptação**.** 2004, 21 f. Proposta de doutoramento (Curso de Doutorado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[SOL 2002] DARTMOUTH SOLAR. **A Scalable Context-Fusion Network for Perva-sive Computing**. Disponível em: <http://www.cs. dartmouth.edu/ ~solar/>. Acesso em: jun. 2002.

[STE 90] STEVENS, W. R. **Unix Network Programming**. Englewood Cliffs: Prentice-Hall, 1990.

[STE 99] STEENKISTE, P. Adaptation Models of Network-aware Distributed Computations. In: INTERNATIONAL WORKSHOP CANPC99, 3., 1999. Orlando, Florida, USA. **Proceedings…** New York: Springer-Verlag, 1999. (Lecture Notes in Computer Science, 1602).

[STI 99] STEIN, B. **Visualization interactive et extensible de programmes paraléles à base de processus légers**. 1999, 148 p. Thesis (Doctoral Programm) - Université Joseph Rourier, Grenoble.

[STO 2002] STOREY, M.; BLAIR, G.; FRIDAY, A. MARE: resource discovery and configuration in ad hoc networks. **ACM Mobile Networks and Applications**, New York, v.7, n.5, p. 377–387, Oct. 2002.

[SUN 2002] SUN, Y.; PERKINS, C. Internet Connectivity for Ad hoc Mobile Networks. **International Journal of Wireless Information Networks**, Berlin, v.9, n.2, p.75-88, Apr. 2002.

[SVA 2001] SILVA JUNIOR, E.; YAMIN, A.; AUGUSTIN, I.; BARBOSA, J.; GEYER, C. Hierarquia de Gerenciamento de Redes com Componentes Móveis. CONGRESO ARGENTINO DE CIENCIAS DE LA COMPUTACIÓN, CACIC, 8., 2001, Santa Cruz, Argentina. **Anales…** Santa Cruz, 2001. 1 CD-ROM.

[TAN 2002] TAN, G. **Self-organizing Bluetooth Scatternets**. Massachusetts: Departament of Eletrical Engineering and Computer Science, 2002. 73p. Dissertation (Master of Science in Computer Science and Engineering) - Massachusetts Institute of Technology, Massachusetts, USA.

[TAN 2003] TANENBAUM, A. **Computer networks**. 4th ed. Saddle River: Prentice Hall, 2003.

[TAV 2002] TAVARES FILHO, J. **EXEHDA-NS:** Uma Contribuição à Mobilidade de Entes no Holoparadigma. 2002. 57 p. Projeto de Diplomação (Bacharelado em Ciência da Computação) – Escola de Informática, UCPEL, Pelotas.

[TMC 2002] IEEE Transactions in Mobile Computing. Disponível em: <http://www.computer.org/tmc/>. Acesso em: out. 2002.

[TON 2002] IEEE/ACM Transactions on Networking. Disponível em: <http://www.comsoc.org/pubs/jrnal/trasnet.html>. Acesso em: jul. 2002.

[TUX 2002] BEA TUXEDO, **A Proven Platform for Distributed Transaction Processing**. Disponível em: <http://www.bea.com/*framework*.jsp>. Acesso em: ago. 2002.

[TYM 98] TYMA, P. Why are we using Java again? **Communications of the ACM**, New York, v.41, n.6, p.38-42, jun. 1998.

[VAH 98] VAHDAT, A. et al. WebOS: Operating System Services for Wide Area Applications. In: IEEE INTERNATIONAL SYMPOSIUM ON HIGH PERFORMANCE DISTRIBUTED, 7., 1998, Chicago. **Proceedings...** Washington: ACM Press, 1998.

[VAR 98] VARGAS, P. **Exploração de Paralelismo OU em uma Linguagem em Lógica com Restrições**. 1998. 95 f. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[VAR 2000] VARSHNEY, U.; VETTER, R. Emerging Mobile and Wireless Networks. **Communications of the ACM**, New York, v.43, n.6, p.73-81, June 2000.

[WAN 2001] WANT, R.; SCHILIT,B. Expanding the Horizons of Location-Aware Computing. **IEEE Computer**, Los Alamitos, v.34, n.8, p.31-34, Aug. 2001.

[WER 94] WERNER, O. **Uma Máquina Abstrata Estendida para o Paralelismo E na Programação em Lógica.** 1994. 145f. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[WIF 2002] WIFI-802.11 Planet. Maximizing Wireless LAN Performance. Disponível em: <http://www.80211-planet.com/>. Acesso em: out. 2002.

[WOL 99] WOLSKI, R.; SPRING, N.; HAYES, J. The Network Weather Service: A distributed resource performance forecasting service for metacomputing, **Journal of Future Generation Computing Systems**, [S.l.], v.15, n.5, p. 757-768, Oct. 1999.

[YAM 93] YAMIN, A.; WERNER, O.; GEYER, C. Prolog Paralelo em Rede de Computadores. In: SIMPÓSIO BRASILEIRO DE ARQUITETURA DE COMPUTADORES - PROCESSAMENTO DE ALTO DESEMPENHO, 5., 1993, Florianópolis. **Anais**... Florianópolis: SBC, 1993. 775p. p.290-303.

[YAM 94] YAMIN, A.; WERNER, O.; BARBOSA, J.; GEYER, C. OPERA Project: an Approach Towards Parallelism Explotation on Logic Programming. In: LOGIC PROGRAMMING WORKSHOP, WLP, 10., 1994, Zurich. **Proceedings...** Zurich:Institut für Informatk der Universität Zürich, 1994. p.29-32.

[YAM 94a] YAMIN, A.; **Um Ambiente Para Exploração de Paralelismo na Programação em Lógica.** 1994. 204 p. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[YAM 99] YAMIN, A. **Um Estudo das Potencialidades e Limites na Exploração do Paralelismo**. 1999. 80 f. Trabalho Individual (Doutorado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[YAM 2000] YAMIN, A. **Escalonamento em Sistemas Paralelos e Distribuídos**. 2000, 136 f. Exame de Qualificação. (Doutorado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[YAM 2000a] YAMIN, A.; GEYER, C. Paralelismo na Programação em Lógica: Alguns Resultados. In: SIMPÓSIO DE INFORMÁTICA DO PLANALTO MÉDIO, 2., 2000, Passo Fundo. **Anais...** Passo Fundo: Universidade de Passo Fundo. 2000.

[YAM 2001] YAMIN, A.; AUGUSTIN, I.; BARBOSA, J.; SILVA, L.; GEYER, C. Explorando o Escalonamento no Desempenho de Aplicações Móveis Distribuídas. In: WORKSHOP EM SISTEMAS COMPUTACIONAIS DE ALTO DESEMPENHO, 2., WSCAD, 2001, Pirenópolis. **Anais...** Pirenópolis: UNB:SBC, 2001.

[YAM 2001a] YAMIN, A. Escalonamento em Sistemas Paralelos e Distribuídos. In: ESCOLA REGIONAL DE ALTO DESEMPENHO**,** ERAD2001, 1., Gramado. **Anais...** Porto Alegre: SBC:UFRGS:PUCRS. 2001. p. 75-126.

[YAM 2002] YAMIN, A.; AUGUSTIN, I.; BARBOSA, J.; GEYER, C. ISAM: a Pervasive View in Distributed Mobile Computing. In: INTERNATIONAL CONFERENCE ON NETWORK CONTROL AND ENGINEERING FOR QOS, SECURITY AND MOBILITY WITH FOCUS ON POLICY-BASED NETWORKING, NET-CON2002, Paris, France. **Proceedings…** Paris: IEEE/IFIP, 2002.

[YAM 2002a] YAMIN, A.; AUGUSTIN, I.; BARBOSA, J.; SILVA, L.; REAL, R.; CAVALHEIRO, G.; GEYER, C. A *Framework* for Exploiting Adaptation in High Heterogeneous Distributed Processing. In: SYMPOSIUM ON COMPUTER ARCHITECTURE AND HIGH PERFORMANCE COMPUTING, SBAC-PAD, 14., 2002, Vitória, Brasil. **Proceedings…** New York: IEEE Press, 2002.

[YAM 2002b] YAMIN, A.; AUGUSTIN, I.; BARBOSA, J.; SILVA, L.; CAVALHEIRO, G.; GEYER, C. Collaborative Multilevel Adaptation in Distributed Mobile Applications. In: INTERNATIONAL CONFERENCE OF THE CHILEAN COMPUTER SCIENCE SOCIETY, SCCC, 12., 2002, Atacama, Chile. **Proceedings…** New York:IEEE Press, 2002.

[YAM 2003] YAMIN, A.; AUGUSTIN, I.; BARBOSA, J.; SILVA, L.; REAL, R.; CAVALHEIRO, G.; GEYER, C. Towards Merging Context-aware, Mobile and Grid Computing. **International Journal of High Performance Computing Applications**, London, v.17, n.2, p.191-203, 2003.

[YAM 2003a] YAMIN, A.; AUGUSTIN, I.; BARBOSA, J.; SILVA, L.; REAL, R.; GEYER, C. EXEHDA: Um Ambiente de Execução para Adaptação Dinâmica ao Contexto de Aplicações na *Pervasive Computing*. **Cadernos de Informática**, Porto Alegre, v. 3, n. 1, p.115-120, jun. 2003.

[YAM 2004] YAMIN, A.; AUGUSTIN, I.; SILVA, L.; REAL, R.; BARBOSA, J.; GEYER, C. EXEHDA: an adaptive *middleware* for the pervasive computing scenery. Submetido à publicação à Revista da SBC. Special Issue: Adaptive Software Systems. 2004.

[YAM 2004a] YAMIN, A.; AUGUSTIN, I.; SILVA, L.; REAL, R.; BARBOSA, J.; GEYER, C. ISAM: Uma Arquitetura de Software para Pervasive Computing. Aceito para publicação no CLEI 2004. Arequipa, Peru. 2004.

APÊNDICE A - EXEHDA: ASPECTOS DE IMPLEMENTAÇÃO

O objetivo deste apêndice é caracterizar os aspectos inerentes à implementação do EXEHDA. Nesse sentido é apresentada a organização dada ao protótipo, assim como as principais tecnologias utilizadas em seu desenvolvimento.

Apêndice A.1 Metodologia de desenvolvimento

O processo de modelagem e implementação do protótipo do EXEHDA seguiu uma filosofia de desenvolvimento incremental, totalmente integrada ao desenvolvimento de outras frentes dentro do Projeto ISAM, em especial a prototipação da linguagem ISAMadapt.

Nesta perspectiva, em um primeiro momento, discussões em grupo envolvendo a equipe de desenvolvimento amadureceram as descrições dos comportamentos esperados da plataforma pervasiva como um todo, as quais foram, posteriormente, especializadas em aspectos particulares do *middleware* e da linguagem de programação. Tais decisões de projeto encontram-se registradas na forma de documentos-texto, organizados de forma hierárquica e armazenados em um repositório denominado Spec. A manutenção desse repositório de especificações é auxiliada pelo uso da ferramenta de controle de versões CVS (Concurrent Versions System, http://www.gnu.org/software/cvs/). Especificamente com relação ao EXEHDA, no subdiretório Spec/EXEHDA/ encontram-se as especificações das atribuições e comportamentos para cada um dos serviços do *middleware*.

Apêndice A.1.1 Especificação e Modelagem

Em um momento seguinte, um processo de modelagem é realizado com base nas especificações em formato texto. O resultado deste processo, que pode implicar a criação de novos componentes de software ou o refinamento dos já existentes, também fica registrado no repositório Spec, mas agora sob a forma de diagramas UML (Unified Modeling Language, http://www.uml.org/). Desta forma, são definidas as interfaces de programação para cada um dos serviços, assim como classes auxiliares que venham a ser oportunas por aspectos de facilidade de utilização, extensibilidade, encapsulamento ou reuso de código. Nesse sentido, uma atenção especial é dada ao uso de padrões de projeto (*design patterns - A Learning Guide To Design Patterns*, http://www.csc.calpoly.edu/~dbutler/tutorials/winter96/patterns/) na modelagem dos componentes do EXEHDA.

Concluída a modelagem de um dado componente, tipicamente, tem início o processo de prototipação deste. Todavia, nem sempre este é o caso. Em alguns momentos, a modelagem permitiu identificar inadequações de um ou outro aspecto definido na especificação. Nestas situações, uma revisão da especificação foi necessária, causando um retorno à etapa das discussões em grupo envolvendo a equipe de desenvolvimento. Este ciclo iterativo caracteriza um primeiro aspecto do desenvolvimento incremental, seguido no EXEHDA.

Apêndice A.1.2 Prototipação e Testes

Na etapa de prototipação, as definições registradas nos diagramas UML são convertidas em código fonte para posterior compilação. Nesse sentido, uma parte bastante significativa do *middleware* encontra-se codificada em linguagem Java, que posteriormente é compilada para bytecode Java. Contudo, uma parte minoritária do código encontra-se codificada em C++, a qual é parcialmente responsável pelo suporte ao subsistema de monitoração. Nesse sentido, busca-se prover implementações mais eficientes e econômicas com relação à utilização dos recursos computacionais do dispositivo sendo monitorado, as quais requerem uma interação mais direta com o hardware e OS do que a propiciada pela plataforma Java. A integração entre estas duas abordagens ocorre através da interface JNI de Java (Java Native Interface, http://java.sun.com/j2se/1.3/docs/guide/jni/), que permite ao bytecode Java interagir com código nativo da plataforma hospedeira. Adicionalmente, alguns utilitários foram implementados em linguagem Shell Script.

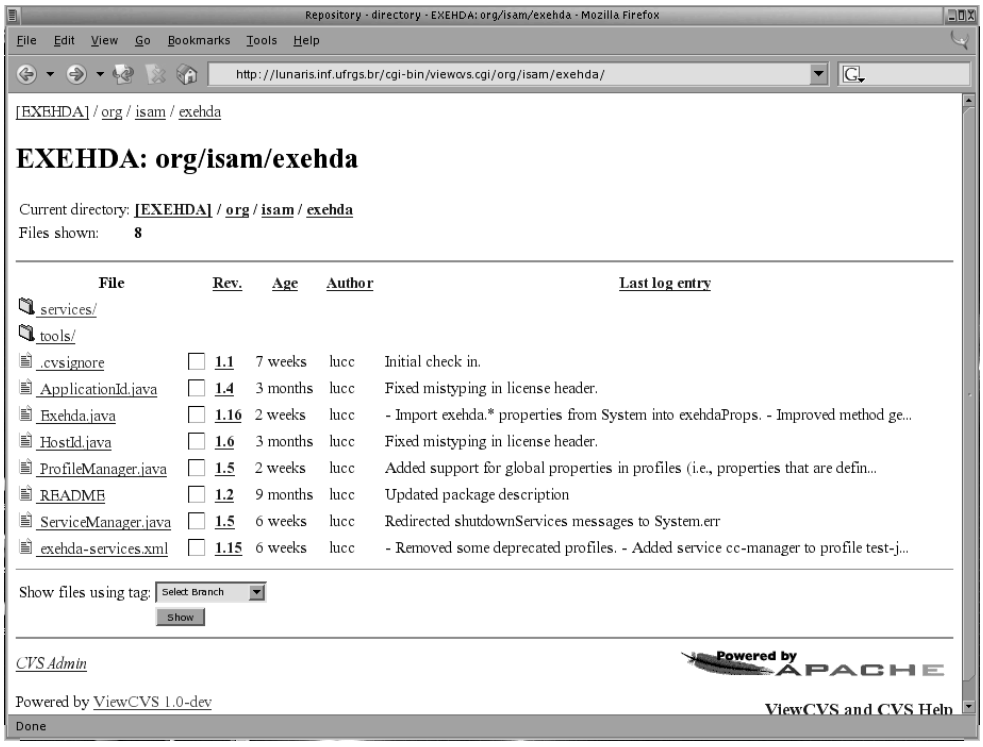


Figura A A.1: Interface Web do Repositório de Código Fonte

No que se refere à documentação do código fonte Java, utiliza-se a ferramenta Javadoc (http://java.sun.com/j2se/javadoc/), a qual permite a geração de documento

hipertexto (html), diretamente a partir do código fonte, contendo descrição das classes Java e da funcionalidade implementada em cada método. Por outro lado, o controle do processo de compilação fica a cargo da ferramenta Ant (Apache Ant, http://ant.apache.org/), a qual disponibiliza uma funcionalidade similar ao tradicional Make do UNIX, porém melhor adaptada para o desenvolvimento de sistemas em Java.

Também o código fonte do protótipo, de forma análoga às especificações, é gerenciado via o sistema de controle de versões CVS. Adicionalmente, no caso do código fonte, existe a possibilidade do acompanhamento do desenvolvimento via interface Web, denominada CVSWeb (FreeBSD CVSweb Project, http://www.freebsd.org/projects/cvsweb.html), conforme ilustrado na figura a a.1.

Um inconveniente do desenvolvimento incremental é que, freqüentemente, a inclusão de uma nova funcionalidade, ou a correção de um *bug* existente, pode modificar a semântica de alguma operação já implementada acarretando a quebra de outra funcionalidade, gerando um efeito cascata. Esta situação é potencializada com o aumento do volume de código fonte.

Com o objetivo de reduzir este problema, adotou-se na implementação do protótipo do EXEHDA uma ferramenta de automação do processo de teste denominada JUnit (http://junit.sourceforge.net/). Nesse sentido, paralelamente à implementação de um dado componente do EXEHDA são também implementados casos de teste com base na especificação daquele componente (texto e UML). Os componentes que implementam os casos de teste são mantidos no repositório CVS que gerencia as versões do código fonte do *middleware*, juntamente com as implementações dos demais componentes do EXEHDA. A ferramenta JUnit permite aplicar em lote todos os casos de teste definidos até o momento para um determinado componente do *middleware*, buscando estressar sua implementação e assim detectar quebras de funcionalidade.

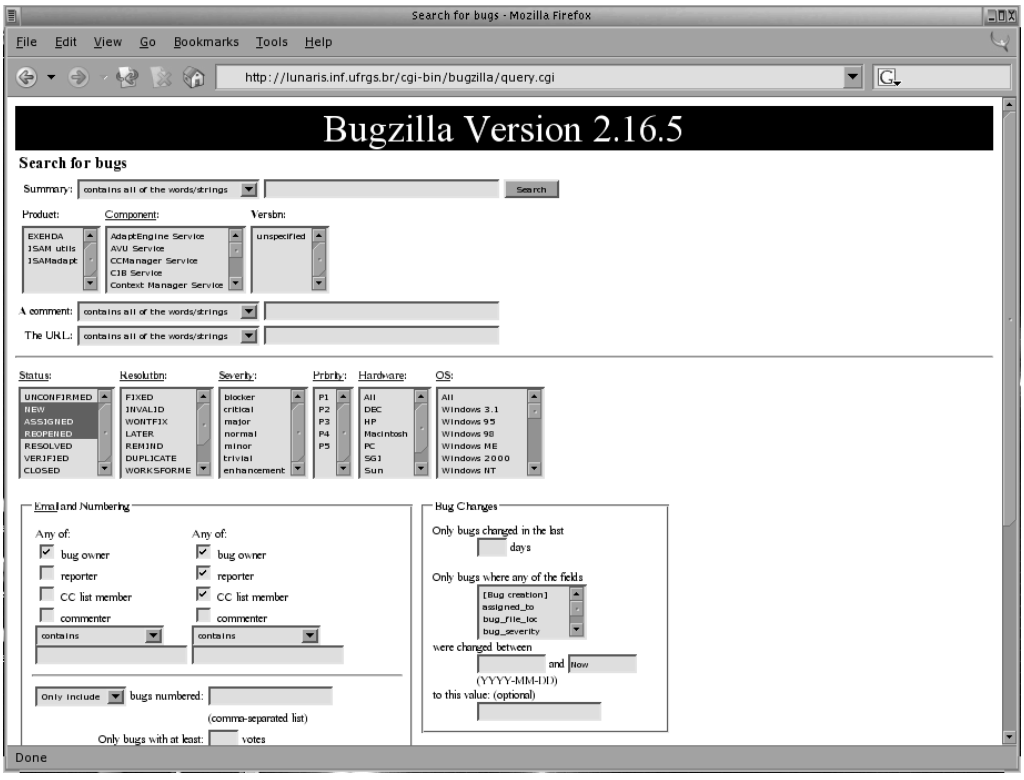


Figura A A.2: Interface do Bugzilla

Observe-se que a adoção da ferramenta JUnit não resolve por completo a questão da validação dos componentes, pois não elimina a necessidade de outros tipos de teste como, por exemplo, execuções efetivamente distribuídas. Nestes casos, existe a necessidade de intervenção do desenvolvedor, ao menos durante a inicialização do caso de teste.

Para organizar o processo de registro e tratamento de *bug*s, adotou-se a ferramenta Bugzilla (http://www.bugzilla.org/). A figura a a.2 apresenta a interface da ferramenta Bugzilla já associada a base de *bugs* do protótipo do EXEHDA. Esta ferramenta é particularmente útil por sistematizar o registro de novos *bugs* e permitir o acompanhamento de sua solução, assim como de qual desenvolvedor a tarefa de resolução de cada um dos *bugs* está associada.

Novamente, na fase de prototipação e testes, a detecção de uma inconformidade do protótipo com relação à especificação inicial, ou o surgimento de novos requisitos, pode realimentar a fase inicial de discussão em grupo, reforçando, assim, a característica incremental do processo de desenvolvimento adotado.

Apêndice A.2 Organização do protótipo

Os pacotes Java que integram a implementação do protótipo estão representados na organização em diretórios apresentada na figura a a.3.

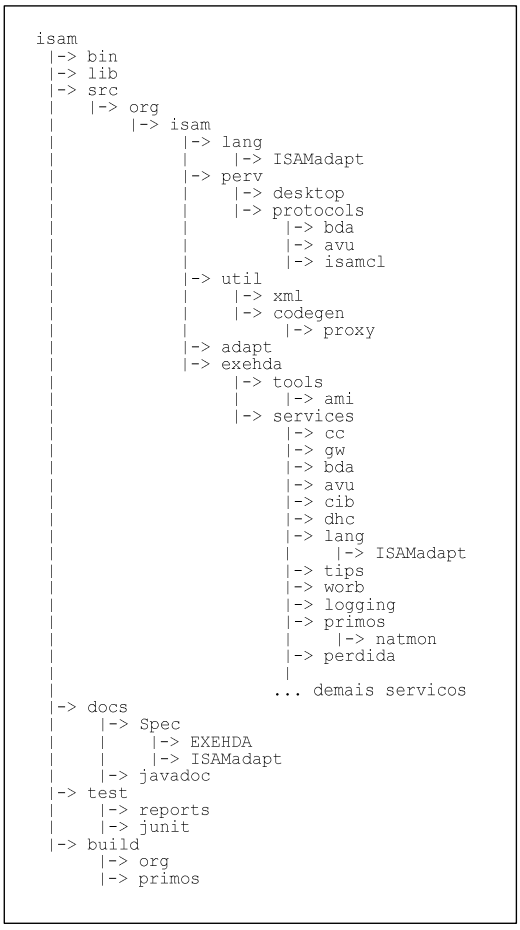


Figura A A.3: Estrutura do protótipo do EXEHDA

Estatísticas gerais do protótipo indicam um total de 222 classes Java (2.9 M Bytes de código fonte), totalizando aproximadamente 25 mil linhas de código, conforme a seguir:

*.java:	134 arquivos (20.231 linhas);
*.cc:	4 arquivos (2.889 linhas);
*.h:	8 arquivos (1.012 linhas).

APÊNDICE B - SISTEMAS DISTRIBUÍDOS E A COMPUTAÇÃO PERVASIVA

Este apêndice tem o objetivo de caracterizar as semelhanças e, sobretudo, as diferenças entre sistemas distribuídos construídos com a premissa de uma infra-estrutura fixa de recursos, e aqueles que dão suporte à mobilidade física de componentes do hardware. A clareza destas diferenças é um aspecto fundamental na concepção do EXEHDA.

Apêndice B.1 Sistemas distribuídos

Para muitos pesquisadores, as questões que surgem em sistemas distribuídos com suporte à mobilidade repetem em grande parte as já existentes nos sistemas distribuídos tradicionais, porém tornam-se potencializadas [SAT 2001]. O que muda efetivamente é a construção das soluções para os problemas, em geral advindos da mobilidade física dos usuários e/ou dispositivos, como a adaptação ao contexto e o acesso pervasivo. Nota-se, também, que o gerenciamento de requisitos não-funcionais, como escalabilidade e segurança, ganha dimensão na Computação Pervasiva, pois a mobilidade física, dentre outros aspectos, introduz um comportamento não-determinístico para procedimentos como a conexão e desconexão às redes existentes.

Esta seção resume o estudo feito nas tecnologias relacionadas quando da etapa de definição das funcionalidades do EXEHDA. Dentre as referências utilizadas neste estudo, destacam-se os trabalhos [SUN 2002, TAN 2003, VAR 2000].

Apêndice B.1.1 Aspectos em sistemas distribuídos na perspectiva do EXEHDA

Considerando o fato de ser a Computação Pervasiva uma proposta de ambiente computacional muito recente [HEN 2001], e, em decorrência, seus conceitos, requisitos e definições não estarem estabelecidos, um dos trabalhos realizados durante a pesquisa do EXEHDA, foi a sistematização das diferenças entre os sistemas distribuídos tradicionais, e aqueles com suporte à mobilidade lógica e física.

Entende-se por sistema distribuído um *framework* integrado que pode ser separado logicamente, e na maioria das vezes fisicamente, em um conjunto de nodos, cada um com o seu estado privado (por exemplo, memória), e com comunicação explícita através de uma mídia (por exemplo, rede) que os interconecta. Os sistemas distribuídos são decorrentes da integração de diversas tecnologias. A necessidade de personalizar

serviços aos usuários conduz à participação de nodos com arquiteturas diferenciadas [COU 2001].

Estes serviços consistem de uma coleção de componentes distribuídos nos nodos. Diversos trabalhos [CHE 2001, COU 2001] já apontaram que a construção de aplicações distribuídas, com suporte ou não à mobilidade física, diretamente sobre a camada de rede do sistema operacional, pode ser trabalhosa, e potencializa a ocorrência de erros de diferentes naturezas; pois, neste caso, os programadores precisam tratar de forma explícita todos os requisitos não-funcionais da aplicação. É importante observar que são comuns requisitos como: balanceamento de carga, replicação, migração de processos etc.

O fato de não serem usados *middlewares* nesses sistemas, além de deixar mais lento o processo de desenvolvimento, compromete aspectos de manutenção e reuso dos componentes de código [RIN 98]. O termo *middleware* é bastante difundido e não existe um consenso sobre o mesmo, sendo empregado para denotar um conjunto genérico de serviços sobre o sistema operacional [IET 2001]. A organização dos sistemas operacionais contemplando o uso de *middlewares* está sintetizada na figura a b.1. É importante ter presente que esta organização é válida para sistemas com ou sem mobilidade física.

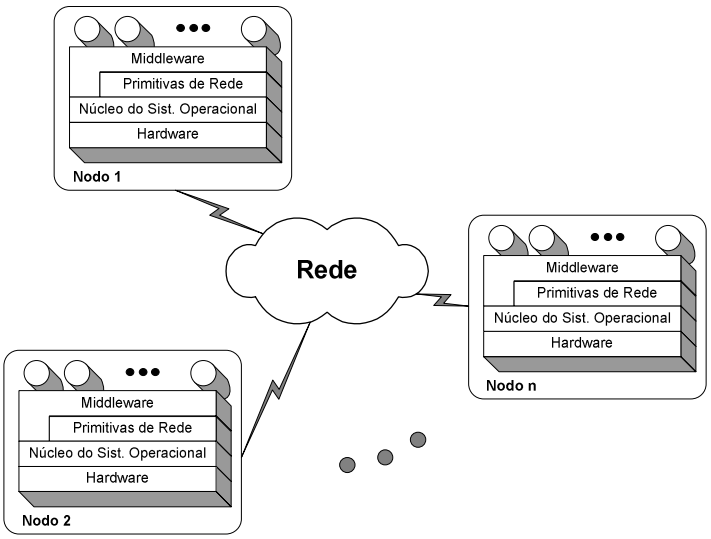


Figura A B.1: Presença do *middleware* em sistemas distribuídos

Vários aspectos resumem as repercussões que a presença da mobilidade física introduz em um sistema distribuído. Entre estes, citam-se:

- **tipo do nodo:** os nodos, em um sistema distribuído sem previsão de mobilidade (nodos fixos), usualmente variam desde equipamentos de uso pessoal (*desktops*), passando por estações de trabalho (*workstations*) indo até computadores de grande porte (*mainframes*). Por sua vez, nodos com mobilidade podem ser PDAs, telefones, ou até mesmo *smartcards*. Deste modo, enquanto que os nodos fixos usualmente são equipamentos dotados de processadores velozes e de boa quantidade de memória, os nodos destinados à operação móvel possuem processadores com pouco poder computacional, pouca

memória, fonte de energia limitada, e uma interface com o usuário com menos recursos;

- **estabilidade da conexão de rede:** em sistemas distribuídos formados por nodos fixos os equipamentos são conectados à rede através de canais permanentes e rápidos. Uma interrupção na operação destes canais tipicamente é decorrente de duas circunstâncias (*i*) uma ação gerencial deliberada, ou (*ii*) uma falha. As interrupções destes canais são consideradas excepcionais na operação rotineira do sistema. Por outro lado, a mobilidade física implica no uso de canais *wireless* (Bluetooth, WaveLAN, HiperLANS etc. [AJI 2000, BLU 2002]) cujo desempenho é fortemente dependente da distância em relação à estação-base (equipamento de gerência) e/ou ao número de nodos que estão na área de cobertura de uma mesma base. A ocorrência de interferências, à medida que o usuário se desloca, pode levar a uma perda total no tráfego de pacotes, e a conexão ser literalmente interrompida. A política de bilhetagem de protocolos para redes-sem-fio (por exemplo, GSM [GSM 2002]) onera os usuários em função do tempo de conexão, levando a uma prática de conexões de menor tempo de duração. Deste modo, seja por falhas operacionais ou por deliberação do usuário, as conexões de rede na computação móvel são intermitentes;

- **dinamicidade do contexto de execução**: em sistemas distribuídos com nodos fixos, as conexões de rede são permanentes e de boa velocidade, a mudança de posição, a inclusão e a remoção de nodos ocorrem em níveis muitos menores que em propostas que contemplam mobilidade. Na Computação Pervasiva, a localização dos serviços existentes é bem mais complexa. O uso de técnicas de difusão, como o *broadcast*, precisa ser cuidadosamente projetado, buscando minimizar o uso dos recursos limitados (consumo da bateria, banda disponível, tempo de conexão etc.). Por outro lado, as condições de trabalho em função da localização são extremamente variáveis. Por exemplo, um PDA com suporte a redes *WiFi* (IEEE 802.11b [WIF 2002]) e GSM [GSM 2002] pode ter sua conexão de rede alternada de 11 Mbps quando localizado nos limites operacionais da empresa, para 9,6 Kbps quando em operação externa. Estes aspectos sinalizam a elevada dinamicidade do contexto de execução quando a mobilidade física é contemplada.

Em função do tipo do nodo, da estabilidade da conexão e da dinamicidade do contexto de execução são possíveis diferentes visões sobre a organização dos sistemas distribuídos. As primeiras considerações, neste sentido, no contexto do Projeto ISAM aparecem nas publicações [YAM 2001] e [AUG 2001a]. Nos trabalhos pertinentes ao EXEHDA são empregadas três visões de referência para sistemas distribuídos: (*i*) sistemas distribuídos tradicionais, (*ii*) sistemas distribuídos híbridos e (*iii*) sistemas distribuídos *ad-hoc*. Os itens a seguir têm por objetivo caracterizar a fronteira conceitual entre cada um destes.

Sistemas distribuídos tradicionais

Os sistemas distribuídos tradicionais são constituídos por uma coleção de nodos fixos, com possibilidade de conexão permanente à rede através de canais rápidos, e apresentam uma configuração cujas variações são poucas no tempo e/ou espaço (estrutura estável, vide figura a b.2) [COU 2001]. Diversos requisitos não-funcionais e independentes de aplicação são perseguidos em um sistema distribuído tradicional. Os principais são:

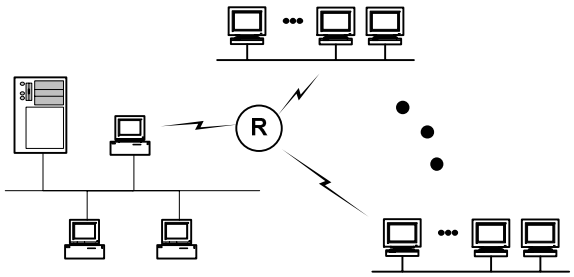


Figura A B.2: Organização de um sistema distribuído tradicional

- **escalabilidade**: é a capacidade do sistema distribuído de crescer para acomodar cargas maiores. A carga pode ser medida por diferentes unidades: número de transações no tempo, número de usuários concorrentes, volume de dados a ser manipulado, etc. A escalabilidade também pressupõe o surgimento de outras necessidades funcionais, o que implica na possibilidade de acomodar novos componentes de hardware e software, preservando a infra-estrutura já existente;
- **tolerância a falhas**: é a capacidade do sistema de se recuperar de falhas sem a necessidade de que a operação do sistema como um todo seja interrompida. As falhas são decorrentes de problemas no software e/ou hardware, e os componentes distribuídos não diretamente envolvidos devem poder manter a sua funcionalidade. Naturalmente, no caso de falhas, algumas vezes serão inevitáveis perdas em alguns aspectos não-funcionais, por exemplo, no desempenho;
- **heterogeneidade**: consiste na integração de componentes programados em diferentes linguagens, executando sob diferentes sistemas operacionais e em plataformas de hardware diversificadas. Na perspectiva atual de sistemas distribuídos de elevada abrangência e com escopo de atuação multi-institucional (por exemplo, Grade Computacional), a convivência com uma elevada heterogeneidade é praticamente inevitável;
- **compartilhamento de recursos**: em um sistema distribuído, recursos de software e hardware (impressoras, bases de informações etc.) podem ser compartilhados pelas aplicações dos usuários. Neste caso, mostra-se indispensável uma disciplina para regular, dentre outros, aspectos de consistência e autorização no acesso aos mesmos.

Apesar dos sistemas distribuídos tradicionais estarem sendo estudados já há aproximadamente 20 anos, a otimização dos seus requisitos funcionais e não funcionais ainda constitui uma frente de pesquisa bastante intensa, a qual totaliza um volume significativo de publicações periódicas, entre elas [TON 2002].

Sistemas distribuídos *ad-hoc*

Este tipo de sistema é constituído por um conjunto de nodos, móveis ou não, que realizam conexões não permanentes através de canais cuja qualidade é extremamente variável. Esta condição caracteriza um ambiente de execução extremamente dinâmico, cuja diferença central em relação aos sistemas distribuídos tradicionais é a não existência de uma infra-estrutura fixa: particularmente os nodos móveis podem se isolar completamente, ou de maneira oportunista formar aglomerados virtuais. As comunicações são simétricas, porém, como dependem da distância, não são transitivas. Tecnologias recentes, concebidas para Computação Pervasiva, como *Bluetooth*, simplificam estas configurações [BLU 2002, TAN 2002]. As redes na nomenclatura *Bluetooth* são dinamicamente formadas por *Piconets* cuja integração constitui as *Scatternets* (vide figura a b.3).

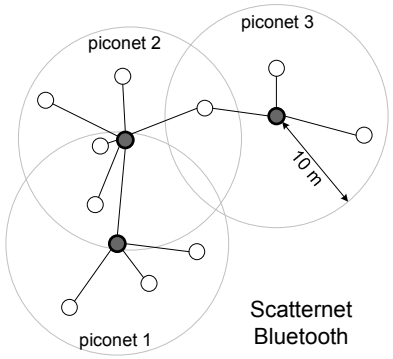


Figura A B.3: Organização de um sistema distribuído *ad-hoc*

A configuração retratada na figura a b.3 caracteriza uma típica *scatternet*, onde pode ser vista a área de abrangência dos transdutores *wireless*, que no caso da tecnologia *Bluetooth* tipicamente é um raio de 10 metros a partir da antena, e também a interligação das *piconets* através de alguns nodos selecionados para operarem como *gateways*. Em alguns casos, os equipamentos *gateway* se caracterizam por um duplo comportamento; atuam como servidores da sua *piconet* e como clientes de outra, constituindo agrupamentos de redes ponto-a-ponto. Uma discussão sobre as possibilidades de configuração de possíveis serviços dos equipamentos *gateways* pode ser encontrada em [TAN 2002].

Na perspectiva *ad-hoc*, os requisitos não-funcionais dos sistemas distribuídos ficam potencializados. Considerando o foco de pesquisa do EXEHDA, três requisitos se destacam:

- **escalabilidade:** três aspectos evidenciam a necessidade de tratar questões relativas a escalabilidade: (*i*) a redução de custo dos PDAs tem tornado os mesmos acessíveis a um número cada vez maior de usuários; (*ii*) a banda passante efetiva em uma célula é fortemente determinada pelo número de nodos que no momento a

compartilham; (*iii*) a presença da mobilidade física dificulta a previsão (avaliação antecipada) do total de nodos nas células;

- **heterogeneidade:** no seu deslocamento, os nodos móveis podem encontrar diferentes tipos de configurações de rede (banda passante, latência etc.), enquanto que equipamentos estacionários se mantêm associados a um determinado tipo;
- **comunicações intermitentes:** enquanto que nos sistemas cabeados as interrupções nas comunicações são consideradas exceções, na proposta *wireless* as mesmas são usuais. Considerando as tecnologias atuais, estas comunicações transientes praticadas na Computação Móvel podem ser vistas como o modo padrão de operação.

No momento em que os recursos físicos participantes do processamento têm seu perfil operacional alterado na presença da mobilidade, as propostas de *middleware* direcionadas para os sistemas distribuídos tradicionais praticamente inviabilizam atingir os requisitos não-funcionais necessários ao gerenciamento do sistema.

As redes *ad-hoc* constituem atualmente o cenário mais moderno em se tratando de infra-estrutura para a computação distribuída com suporte a mobilidade física, e são uma frente de pesquisa muito ativa. Um indicador neste sentido são as diversas publicações sumarizadas em [MAN 2003].

Sistemas distribuídos híbridos

Este tipo de sistema tem uma organização que é intermediária entre as redes somente compostas de computadores fixos (tradicionais), e as direcionadas a computadores móveis (*ad-hoc)*. Sua composição tem por base uma infra-estrutura estática similar àquela dos sistemas distribuídos tradicionais (roteadores, *gateways*, *hubs* etc.) [TAN 2003], tendo no perímetro desta infra-estrutura equipamentos de rede (*base stations*) com suporte para operações *wireless* [VAR 2000], as quais viabilizam a interconexão dos nodos móveis que estão na sua área de cobertura. Uma visão desta organização pode ser vista na figura a b.4.

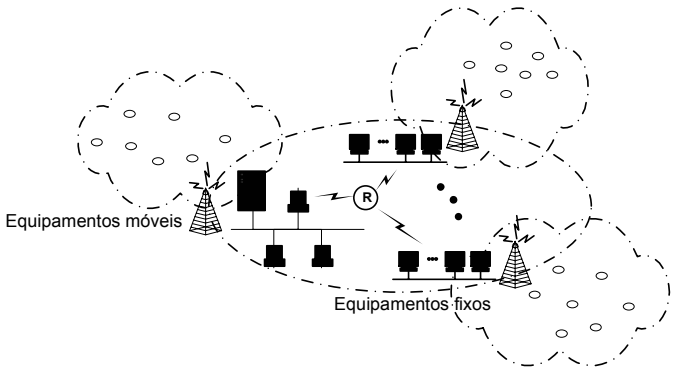


Figura A B.4: Organização de um sistema distribuído híbrido

A proposta híbrida tem tido o seu emprego potencializado em situações nas quais as estratégias de conexão baseadas em cabeamento físico das redes estruturadas não se mostram oportunas frente à demanda por conexões temporárias, como em conferências,

usos militares ou situações de perda temporária de infra-estrutura provocadas por desastres, acidentes, etc.

Por outro lado, sistemas distribuídos híbridos constituem alternativas reais para a integração da vasta gama de serviços disponíveis nos sistemas distribuídos tradicionais, em oposição à flexível e moderna proposta *ad-hoc*. Publicações discutindo aspectos associados a este tipo de sistema estão cada vez mais presentes em publicações indexadas e internacionalmente reconhecidas [RIM 2002, SUN 2002].

Uma instância desta visão de sistemas distribuídos híbridos foi a opção utilizada para compor o ambiente pervasivo ISAM (ISAMpe), descrito na seção 4.1.

As características introduzidas pelos modelos tecnológicos que compõem os três tipos de sistemas distribuídos sumarizados neste item implicam em comportamentos operacionais, os quais serão discutidas a seguir. Uma clareza na expectativa destes comportamentos é fundamental na construção dos argumentos que nortearam a concepção do modelo computacional do EXEHDA.

Apêndice B.1.2 Resumo Comparativo dos *middlewares* em sistemas distribuídos

A análise comparativa para *middlewares* destinados a sistemas distribuídos teve por objetivo subsidiar a discussão sobre os trabalhos relacionados ao EXEHDA (capítulo 2). A mesma tem por base três parâmetros: (*i*) o custo computacional que o *middleware* introduz, (*ii*) quais os paradigmas de comunicação que este disponibiliza e (*iii*) que forma de representação de contexto é empregada. Os itens a seguir contextualizam os parâmetros empregados.

Custo computacional

O custo computacional que um *middleware* introduz nos nodos do sistema distribuído é dependente dos requisitos exigidos pelo ambiente de execução, tanto pelos aspectos quantitativos como pelos aspectos qualitativos dos serviços que devem ser disponibilizados.

Da revisão feita nas áreas de conhecimento que integralizam esta proposta de tese, apresentadas na introdução (capítulo 1), podem ser extraídos dois exemplos que bem caracterizam esta situação:

- considerando o serviço de alocação de tarefas aos recursos computacionais existentes (escalonamento), os estudos feitos por Yamin [YAM 2001a, YAM 2001 e YAM 2002] ratificaram que, dependendo do número de variáveis a ser considerado para tomada de decisão, o custo global introduzido, tanto em processamento como em ocupação dos canais de comunicação pela troca de dados para tomada de decisão, pode ser bastante elevado. Com facilidade, a complexidade dos algoritmos pode atingir patamares computacionalmente não-tratáveis no tempo desejado;
- por sua vez, considerando um serviço como a replicação [FER 2001], bastante usual no suporte à tolerância a falhas, ou para tratamento da escalabilidade, dependendo da estratégia utilizada para manutenção do sincronismo entre as cópias, a operação pode

demandar recursos significativos do sistema; por exemplo, a banda-passante da rede.

Estes exemplos evidenciam que, dependendo das características do sistema distribuído, uma determinada organização do *middleware* poderá ser mais oportuna que outra, tanto no que diz respeito aos recursos de infra-estrutra disponíveis como dos requisitos que devem ser disponibilizados.

Esta constatação foi determinante para a opção de conceber o EXEHDA com comportamento reflexivo na adaptação ao contexto, e com a capacidade de receber diretrizes da aplicação para nortear as políticas de adaptação.

Paradigmas de comunicação

Existem dois paradigmas de comunicação, em termos gerais, que um *middleware* pode implementar: síncronos e assíncronos [TAN 2003]. O primeiro pressupõe, para sucesso da operação, que os participantes, no momento da comunicação, estejam simultaneamente operacionais e interligados. Por sua vez, o paradigma assíncrono não exige que o transmissor e receptor estejam operacionais ao mesmo tempo.

Entre estes dois extremos existem várias alternativas; destacam-se duas: (*i*) *Oneway*, que devolve o controle do fluxo de execução ao cliente sem aguardar que o servidor termine o processamento solicitado. Isto somente será possível quando a semântica operacional do cliente não depender do resultado que está sendo produzido; (*ii*) *Deferred Synchronous*, que também retorna o controle imediatamente para o cliente, porém tem a responsabilidade de voltar a sincronizar com o servidor para recolher o resultado. Caso o servidor não esteja pronto para responder nesta segunda conexão, o cliente ficará bloqueado, aguardando o servidor atendê-lo [COU 2001].

A perspectiva cada vez maior da utilização de ambientes distribuídos de larga escala vem potencializando as discussões a respeito de alternativas para mecanismos de comunicações [GET 2002]. Considerando a premissa do projeto ISAM de trabalhar em um ambiente de rede global, a definição da gerência das comunicações foi aspecto importante na definição do EXEHDA.

Representação de contexto

Considerando o cenário de pesquisa do EXEHDA, outro parâmetro para diferenciar as classes de *middlewares* está associado à forma como são tratadas as informações correspondentes ao meio em que está ocorrendo a execução. Estas informações, ou parte delas, poderão ser compartilhadas com as camadas superiores do *middleware,* atingindo a própria API disponibilizada ao usuário. Quando ocorre desta forma, fica caracterizada a possibilidade de que as informações de contexto sejam exploradas (*context-awareness*) nas aplicações em execução. Em [CHE 2001, KOR 2001] estão discutidos os principais trabalhos referentes a este tema.

A disseminação da Computação Pervasiva aponta na direção que as aplicações *context-aware* sejam também distribuídas, isto é, que os componentes passíveis de se adaptarem ao contexto estejam localizados em diferentes nodos do sistema [AUG 2004, YAM 2003].

O EXEHDA tem na sua concepção a premissa que as informações sobre o contexto também poderão ser empregadas pelo *middleware* de forma transparente para as aplicações. Deste modo, fica flexibilizado que tanto a aplicação como o ambiente de

execução tenham participação ativa no gerenciamento das adaptações oportunizadas durante o processamento [YAM 2002a, YAM 2002b].

No Quadro A2.1, a seguir, estão sumarizadas as inter-relações entre os aspectos levantados nesta seção.

Parâmetros	**Tipos de Sistemas distribuídos**	
	Sistemas distribuídos tradicionais	**Sistemas distribuídos ad hoc e híbridos**
Custo computacional	Composto por nodos fixos, os quais implementam serviços com elevado custo computacional.	Presença de computadores móveis, o que implica em que o núcleo do *middleware* deve ser leve. Deve ser prevista uma instalação seletiva de serviços nos nodos. Se necessário os nodos fixos receberão incumbências de maior custo computacional.
Paradigma de comunicação	São empregadas conexões permanentes, sobre as quais ocorrem comunicações síncronas.	Ocorrência de conexões intermitentes, o que exige suporte para comunicações assíncronas.
Representação do contexto	Pequena dinamicidade do contexto de execução (praticamente estático), o que faculta transparência no mecanismo de adaptação (o *middleware* pode gerenciar sózinho a adaptação).	Elevada dinamicidade do contexto de execução, a qual não permite previsão sobre a disponibilidade de recursos. Esta dinamicidade implica na participação da aplicação na tomada de decisão (uso do conhecimento do usuário), a qual, assim como o *middleware*, deve neste caso ter consciência do contexto em que a execução está ocorrendo.

Quadro AB.1 - Classificação em sistemas distribuídos

APÊNDICE C - O EMPREGO DE JAVA NO EXEHDA

Java é um dos componentes da arquitetura de software projetada para o ISAM (vide seção 3.4). A presença da linguagem Java no âmbito dos trabalhos pertinentes ao EXEHDA se dá sob duas ópticas:

- o código ISAMadapt é traduzido para Java [AUG 2003];
- a maioria dos serviços do EXEHDA é programada em Java [SIL 2003, REA 2004, YAM 2003].

Aspectos sobre a tradução do código ISAMadapt para Java podem ser encontrados em [AUG 2004, BAR 2002, BAR 2001]. Neste apêndice estão resumidos os principais aspectos pertinentes ao emprego de Java na implementação dos serviços do EXEHDA. Ressalte-se que alguns serviços, particularmente os relacionados a aspectos de monitoração, por motivo de desempenho são compilados para código nativo da plataforma em que irão atuar.

Apêndice C.1 Motivações para o uso da plataforma Java

Os principais aspectos da linguagem Java que a tornam oportuna como plataforma para prototipação dos serviços do EXEHDA, são:

- portabilidade;
- suporte à carga dinâmica de código;
- segurança;
- suporte à concorrência e sincronização;
- produtividade no desenvolvimento estruturado de software e
- existência de um modelo de computação distribuída baseado em objetos – RMI - bastante difundido.

Uma revisão destes aspectos está feita a seguir.

Apêndice C.1.1 Portabilidade

Do ponto de vista do processamento distribuído, a independência de plataforma é a característica mais interessante de Java. Esta independência decorre de dois aspectos: (*i*) da disponibilização de uma vasta API padronizada, mas principalmente, (*ii*) da utilização de uma representação de código executável baseada em um formato de instrução portável, denominado bytecode [LIN 97].

Em geral, um programa Java é composto por um conjunto de classes. Pelo processo de compilação, o código fonte das classes Java é convertido para uma representação no formato de bytecodes. Essa seqüência de bytecodes não é executada diretamente como um programa compilado para código nativo, mas demanda a utilização de um elemento intermediário que apresente uma interface uniforme sobre as diferentes arquiteturas de hardware existentes. Tal elemento, no *framework* Java, é a Máquina Virtual Java (JVM) [LIN 97].

A abordagem de máquina virtual transfere a questão da manutenção da portabilidade das aplicações para os implementadores da JVM, a qual, sim, deve ser implementada e compilada especificamente para cada plataforma alvo a ser suportada. Atualmente, existem portes da JVM para plataformas como Solaris, Linux, Windows, MacOS, além de implementações específicas para sistemas embarcados, PDAs, entre outras. Esta ampla cobertura de plataformas provida por Java, é um aspecto significativo na proposta do EXEHDA.

Apêndice C.1.2 Carga dinâmica de código

O processo de carga de classes é feito de forma dinâmica na JVM, sendo controlado por um carregador de classes (ou *Class Loader* na denominação original do *framework* Java). O carregador de classes é um objeto plugável do sistema, podendo ser personalizado de forma a implementar a carga de classes tanto a partir do sistema de arquivos da máquina, ou de um servidor HTTP na Internet, quanto de um banco de dados, por exemplo.

Apêndice C.1.3 Segurança

A JVM executa programas Java, interpretando instrução-a-instrução ou por meio de compilação *Just-in-Time*, num ambiente controlado conhecido como *Sandbox*. Esse ambiente implementa uma forte verificação dinâmica sobre as classes da aplicação Java, de forma a evitar acessos indevidos de memória, que poderiam implicar em falhas de segurança. Aliado a isso, a plataforma Java provê em sua API acesso controlado aos recursos do sistema nativo (sistema de arquivos, *sockets*) por meio de gerentes de segurança (*Security Managers* na nomenclatura de Java). A combinação desses dois elementos permite a utilização do mecanismo de carga dinâmica de código fornecido por Java em ambientes distribuídos, sem que isto implique um comprometimento da segurança dos EXEHDAnodos que alojam uma JVM.

Apêndice C.1.4 Concorrência e sincronização

Sob o ponto de vista do processamento paralelo, o atrativo de Java está na presença de construções nativas à linguagem para a expressão e controle da concorrência. Java disponibiliza em sua API a classe Thread, a qual serve para disparar linhas de execução concorrentes dentro da aplicação. Esta classe é extensamente utilizada já na própria API padrão do sistema, na implementação de funcionalidades como o modelo de eventos que suporta as classes relativas à construção de interfaces gráficas.

O modelo de monitores é utilizado por Java para controle da concorrência no acesso aos objetos. Construções explícitas, como *threads*, podem não representar o mais elevado grau de abstração na expressão do paralelismo entre as arquiteturas de

processamento paralelo existentes. No entanto, o fato de tal construção estar presente na constituição básica da linguagem constitui um enorme avanço no sentido de aumentar a portabilidade das aplicações paralelas, quando comparado a linguagens tradicionais como C/C++, que suportam esse tipo de construção apenas por meio de bibliotecas, as quais muitas vezes são específicas para determinadas plataformas. É garantido que toda plataforma para a qual houver um porte de JVM, aplicações desenvolvidas usando as *threads* Java poderão ser executadas sem recompilação.

Apêndice C.1.5 Produtividade no desenvolvimento estruturado de software

A seguir, são enumerados alguns aspectos de Java que tendem a beneficiar o processo de desenvolvimento de software como um todo [TYM 98]:

- independência de plataforma: reduz os custos de desenvolvimento e manutenção, pois unifica a implementação em uma única versão de código, a qual é diretamente executável em todas as plataformas que suportam a tecnologia Java. Nesse sentido, Java provê ainda uma vasta API gráfica denominada *Swing*, além de extensões para manipulação de imagens 2D e 3D;
- produtividade: o gerenciamento automático de memória provido pelo mecanismo de coleta de lixo, somado ao fato de Java ser uma linguagem fortemente tipada, além de não permitir o uso de aritmética de ponteiros e empregar um tratamento estruturado de exceções por meio de blocos `try/catch`, tendem a reduzir o tempo gasto em depuração, visto que tornam a programação menos suscetível a erros;
- componentes plugáveis (*Java Beans*): tendência a simplificar e, por conseguinte, aumentar o reuso de código;
- linguagem de programação nova: Java apropriou-se de características interessantes de outras linguagens mais tradicionais como C++, Smaltalk, e suprimiu outras que tendiam a dificultar a programação. Como resultado, Java é uma linguagem com tendência a ter menos “retificações corretivas”;
- orientação a objetos: modelo adequado ao particionamento da aplicação, favorecendo o encapsulamento de dados e componentes. Embora não tão versátil como o modelo de herança múltipla empregado em C++, o modelo de herança única e interfaces, utilizado em Java, justamente por ser mais simples, tende a favorecer o entendimento do código já desenvolvido; e
- similaridade com C/C++: a torna familiar e de fácil aprendizado dada a grande difusão da linguagem C.

Apêndice C.1.6 Java RMI: computação distribuída baseada em invocações remotas de métodos

Outro aspecto oportuno de Java, no tocante à computação distribuída, é a existência de um modelo de computação baseado em objetos distribuídos, denominado RMI (*Remote Method Invocation*) [FAR 98]. No modelo RMI, os objetos Java passam a estar habilitados a receberem invocações de métodos advindas de outros nodos, ou seja, existe todo um suporte à execução de invocações remotas de método.

RMI trata transparentemente, por meio do uso de *stubs* (*proxys*) e da API de reflexão de Java, o processo de *marshaling*[3] de argumentos [BIR 96], para execução da chamada de método no objeto destino e posterior retorno de resultados. Do ponto de vista do programador, a sintaxe utilizada segue o mesmo modelo aplicado às invocações locais de método, uniformizando o código fonte e facilitando o entendimento da semântica do programa.

O processo de invocação remota não se dá, no entanto, de forma totalmente transparente, mesmo utilizando-se RMI. Deve-se isso à inclusão de novas variáveis no sistema, como a heterogeneidade das latências e a susceptibilidade a erros de desconexão da comunicação em rede. Portanto, faz-se necessário algum tipo de tratamento de exceções adicional, devido à possibilidade de interrupções na comunicação, o qual antes não era preciso quando de invocações locais de método. Este aspecto tem particular significado no contexto do EXEHDA.

A abstração provida por RMI mostra-se oportuna, pois preserva, no âmbito de sistemas distribuídos, o paradigma orientado a objetos nativo de Java, uniformizando o processo de desenvolvimento de software distribuído.

Apêndice C.2 Insuficiências da plataforma Java

Considerando as necessidades operacionais do EXEHDA, dentre as características atuais da plataforma Java que comprometem seu uso de forma direta na programação de seus serviços, destacam-se:

- a inexistência de um mecanismo de distribuição automática dos objetos a serem instanciados ao longo dos nodos do sistema distribuído, ou mesmo de um mecanismo de criação remota ou de migração de objetos;
- a inexistência de um suporte à monitoração da carga nos nodos, bem como das atividades desempenhadas pelos objetos de uma aplicação, de forma integrada à plataforma;
- a associação da implementação atual de RMI a um protocolo orientado a conexão (i.e., a conexão precisa ser mantida durante toda a execução de uma invocação remota de método), baseado em TCP/IP.

Das duas primeiras limitações decorre a dificuldade em se implementar procedimentos adaptativos funcionais (troca de objeto em execução) ou não funcionais (como balanceamento de carga).

[3] Representação em um formato que possa ser eficientemente interpretado pelo destino da chamada remota.

Por outro lado, a abordagem atualmente utilizada na implementação das comunicações RMI possui dois grandes inconvenientes: (*i*) a ineficiência na presença de hardware otimizado de comunicação e (*ii*) a difícil construção de semânticas de execução adequadas para aplicação distribuída em ambientes onde os nodos estão sujeitos a desconexões freqüentes, como é o caso dos ambientes pervasivos.

Tais limitações de Java têm implicações diretas na concepção da arquitetura de software provida pelo EXEHDA e, por conseguinte, foram contempladas nos esforços de pesquisa desenvolvidos.