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

IARA AUGUSTIN

Abstrações para uma Linguagem de Programação Visando Aplicações Móveis em um Ambiente de *Pervasive Computing*

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

Prof. Dr. Cláudio Fernando Resin Geyer
Orientador

Porto Alegre, janeiro de 2004

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Augustin, Iara

Abstrações para uma Linguagem de Programação Visando Aplicações Móveis em um Ambiente da *Pervasive Computing* / Iara Augustin – Porto Alegre: Programa de Pós-Graduação em Computação, 2004.

194 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. *Pervasive Computing*. 2. Computação Móvel. 3. Paradigmas de Programação. 4. Aplicações Conscientes do Contexto. I. Geyer, Cláudio Fernando Resin. II. Título.

À minha mãe, irmã, e meus pequenos sobrinhos,
por aceitarem a minha ausência.

AGRADECIMENTOS

Agradeço a todos que direta ou indiretamente contribuíram para este trabalho: professores, colegas e funcionários do Instituto de Informática. Em especial agradeço

ao meu amigo Adenauer, por aceitar seguir junto comigo este caminho.

ao Luciano o apoio, dedicação e sincera disposição em ajudar, e as esclarecedoras “discussões” dos últimos meses. Tens minha admiração pela pessoa que és.

ao Rodrigo, a amizade e as conversas animadas.

à Márcia, a amizade, carinho e apoio logístico.

ao Gustavo, por contribuir na implementação do WalkEd.

aos bolsistas Rafael Pereira Pires e Ricardo Redin, UFSM, por contribuírem na implementação do ISAMadapt.

ao meu orientador, Cláudio Geyer, a amizade e disposição em atender meus pedidos, e por contribuir com a criação de um grupo de estudos que continuará a existir além dessa tese.

Agradeço também à FAPERGS o apoio financeiro ao projeto ISAM, e ao CNPq/FINEP/SEPIN os recursos concedidos ao projeto contextS.

Por fim, agradeço à inspiração e à intuição a escolha do tema.

As pessoas entram em nossas vidas por acaso,
mas não por acaso permanecem.

SUMÁRIO

LISTA DE ABREVIATURAS

API	*Application Programming Interface*
AVA	Ambiente Virtual da Aplicação
AVU	Ambiente Virtual do Usuário
BNF	*Backus-Naur Form*
CDC	*Connected Device Configuration*
CLDC	*Connected Limited Device Configuration*
CORBA	*Common Object Request Broker Architecture*
CSCW	*Computer Supported Cooperative Work*
EXEHDA	*Environment Execution for High Distributed Applications*
ISAM	Infra-estrutura de Suporte às Aplicações Móveis Distribuídas
ISAMbda	Base de Dados Pervasiva
ISAMpe	*ISAM Pervasive Environment*
J2ME	Java *Micro Edition*
J2SE	Java *Standard Edition*
LDAP	*Lightweight Directory Access Protocol*
MIDP	*Mobile Information Device Profile*
P2P	*Peer-to-Peer Computing*
PRIMOS	*Primitives for Objects Scheduling*
PvC	*Pervasive Computing*
PDA	*Personal Digital Assistant*
QoS	Qualidade de Serviço (*Quality of Service*)
RMI	*Remote Method Interface*
RPC	*Remote Procedure Call*
SNMP	*Simple Network Management Protocol*
TIPS	*Tips Is a Probabilistic Scheduling*
UML	*Unified Modelling Language*
URL	*Universal Resource Locator*
XML	*Extended Modelling Language*

LISTA DE FIGURAS

LISTA DE QUADROS

RESUMO

Computação Móvel é um termo genérico, ainda em definição, ao redor do qual se delineia um espectro de cenários possíveis, desde a Computação Pessoal, com o uso de computadores de mão, até a visão futurista da Computação Ubíqua. O foco do projeto ISAM (Infra-estrutura de Suporte às Aplicações Móveis Distribuída), em desenvolvimento no II/UFRGS, é a *Pervasive Computing*. Esta desenha um cenário onde o usuário é livre para se deslocar mantendo o acesso aos recursos da rede e ao seu ambiente computacional, todo tempo em qualquer lugar. Esse novo cenário apresenta muitos desafios para o projeto e execução de aplicações. Nesse escopo, esta tese aprofunda a discussão sobre questões relativas à adaptação ao contexto em um ambiente *pervasivo* sob a ótica de uma Linguagem de Programação, e define uma linguagem chamada ISAMadapt.

A definição da linguagem ISAMadapt baseia-se em quatro abstrações: contexto, adaptadores, políticas e comandos de adaptação. Essas abstrações foram concretizadas em duas visões: (1) em tempo de programação, através de comandos da linguagem e arquivos de configuração, descritos com o auxílio do Ambiente de Desenvolvimento de Aplicações; (2) em tempo de execução, através de serviços e APIs fornecidos pelos componentes que integram o ambiente de execução *pervasiva* (ISAMpe). Deste, os principais componentes que implementam a semântica de execução da aplicação ISAMadapt são: o serviço de reconhecimento de contexto, `ISAMcontextService`, e a máquina de execução da adaptação dinâmica, `ISAMadaptEngine`.

As principais contribuições desta tese são: (a) primeira linguagem para a codificação de aplicações *pervasivas*; (b) sintaxe e semântica de comandos para expressar sensibilidade ao contexto *pervasivo*; (c) fonte para o desenvolvimento de uma metodologia de projeto de aplicações *pervasivas*; (d) projeto ISAM e o projeto contextS (www.inf.ufrgs.br/~isam) que fornecem suporte para o ciclo de vida das aplicações, desde o desenvolvimento até a execução de aplicações *pervasivas*.

Palavras-Chaves: *Pervasive Computing*. Computação Móvel. Aplicações *Pervasivas*. Aplicações Conscientes do Contexto. Paradigmas de Programação. Linguagens de Programação.

Programming Language Abstractions for Coding Mobile Applications at the Pervasive Computing Environment

ABSTRACT

Mobile Computing is a generic term, yet to be defined, under its scope there is a range of possible scenarios, from Personal Computing to the futuristic vision of Ubiquitous Computing. The focus of ISAM project, carried out at II-UFRGS, is the Pervasive Computing. It designs a scenario where the user is free to move anywhere, all the time, maintaining the network access and having access to his/her virtual computing environment. This new scenario poses many challenges to the design and execution of applications. It is within this scope that my thesis deepens the discussion on many issues involved in the context adaptation in a pervasive environment under the programming language viewpoint. It defines a programming language called ISAMadapt. The language definition is based on four abstractions: context, adapters, adaptation policies and commands. These abstractions were rendered in two visions: (1) at the programming time, through language commands and configuration files described with the help of the ISAMadapt Interface Development Environment; (2) at the run-time, through services and APIs provided by components of the ISAM Pervasive Environment (ISAMpe). The main components that implement the execution semantic of the ISAMadapt pervasive applications are: ISAMcontextService, the context recognition service, and ISAMadaptEngine, the dynamic adaptation engine. The main contributions of this thesis are: (a) the first programming language for pervasive applications; (b) syntax and semantics of commands to express context-aware adaptation; (c) source for development of a methodology to design pervasive applications; (d) contextS and ISAM projects (www.inf.ufrgs.br/~isam), that provide the support for the pervasive applications life cycle, from development to execution.

Keywords: Pervasive Computing. Mobile Computing. Pervasive Applications. Context-aware Mobile Systems. Programming Paradigms. Programming Language.

1 INTRODUÇÃO

A Computação Móvel está adquirindo maior popularidade à medida que dispositivos móveis tornam-se disponíveis e oferecem facilidades de uso para seus usuários (Interactions, 2002). Prevê-se que a mobilidade tornar-se-á uma das características-chave dos sistemas de computação. Nos últimos anos, cresceu o interesse da indústria e da academia pelo estudo da mobilidade. Isto pode ser comprovado pelo número de trabalhos que surgiram na literatura neste início de década, pela abertura da área de Computação Móvel nos congressos (redes, sistemas distribuídos, sistemas de informação e engenharia de software), e pelo lançamento de periódicos como *Pervasive Computing, Mobile and Ubiquitous Computing* (IEEE, primeira edição em março/2002) e *IEEE Transaction on Mobile Computing* (lançado em 2002). Por acreditar que a Computação Pervasiva será o paradigma do século 21, escolheu-se esta como a área tema dessa tese.

1.1 Área Tema: a Computação *Pervasiva*

Um dos pioneiros na área foi Mark Weiser com sua proposta de Computação Ubíqua (onipresente, *Ubiquitous Computing*) (WEISER, 1991; WEISER, 1993) que visava tornar a computação integrada na vida do usuário. Essa visão enfatiza a invisibilidade da computação e a centralização desta nas tarefas do usuário, onde o computador atua em *background* e ajusta-se às necessidades dos usuários, tornando-se a "tecnologia que desaparece", em oposição à situação atual, onde o usuário deve aprender a lidar com o computador e a gerenciar suas aplicações. Outras propostas derivadas desta linha de pensamento são a Computação Invisível de D. A. Normam (1998), a era Post-PC da Sun (www.sun.com/j2me), e a Computação Pró-ativa da Intel (TENNENHOUSE, 2000). Uma visão mais operacional, a qual parece estar se consolidando (SAHA, 2003), adota o nome de Computação *Pervasiva*[1] (*Pervasive Computing* - computação espalhada) promovido pela IBM.

Este trabalho adota o termo Computação *Pervasiva* como cenário em que a computação alvo desta proposta se desenvolve. A Computação *Pervasiva* fornece uma visão da computação futura onde o poder computacional estará disponível em qualquer lugar. Este cenário de mobilidade global emprega uma variedade de dispositivos móveis

[1] O termo *pervasivo* não existe na língua portuguesa. Alguns pesquisadores propõem a tradução para ubíquo. Porém considera-se esta tradução inadequada. Na literatura existem os dois termos Ubiquitous Computing e Pervasive Computing. É de nosso entendimento que esses termos não são sinônimos, mas propostas de uma visão futura da computação, a segunda mais próxima que a primeira considerando o momento de desenvolvimento tecnológico atual (ver Apêndice A). Pode-se dizer que a *Pervasive Computing*, associada a outras tecnologias como a computação de vestir (*Wearable Computing*) e redes inteligentes, convergirá para a *Ubiquitous Computing*.

e estáticos que dinamicamente se conectam, reagem ao ambiente corrente, e se coordenam uns com os outros e com os serviços de rede para auxiliar o usuário na realização de suas tarefas. Espera-se que os dispositivos móveis descubram, dinamicamente, outros dispositivos em uma dada localização, e continuem a funcionar ainda que em modo desconectado e sob condições de recursos e serviços limitados. O objetivo primário da Computação *Pervasiva* é fornecer aos usuários um acesso uniforme e imediato a informações e, transparentemente, suportar a execução de suas tarefas. A sua essência é um ambiente saturado com capacidade de computação e de comunicação integradas com as atividades do usuário (SATYANARAYANAN, 2001).

Após uma década de progresso do hardware, muitos elementos críticos da Computação Ubíqua de Weiser, que eram exóticos em 1991, são agora viáveis comercialmente. Telefones celulares incluem processador e armazenamento, PDA's, como Palm's e outros, estão aumentando seu poder computacional, e outros dispositivos, como combinações de PDA's e telefones celulares, começam a aparecer no mercado. Redes sem fio, como o padrão IEEE 802.11 e o Bluetooth (www.bluetooth.com), fornecem conectividade local, e a Internet fornece conectividade global. Segundo Satyanarayanan (2001), está-se agora mais bem posicionado para começar a realizar a visão de Weiser, porém a Computação Ubíqua ainda é uma realidade muito distante (BANAVAR; BERNSTEIN, 2002).

Para que esta visão se torne realidade, desenvolvedores começam a construir a infra-estrutura e a experimentar com aplicações que se tornem úteis aos usuários. Recentemente, grandes projetos estão sendo propostos em grandes universidades, como o projeto Aura na Carnegie Mellon University (www.cs.cmu.edu/~aura), Oxygen no MIT (http://oxygen.lcs.mit.edu) e Endeavour em Berkeley (http://endeavour.cs.berkeley.edu). No centro dessas visões está a computação centrada no usuário (*user-centric*), a tecnologia para usuários e serviços em escala global. No Brasil, destacam-se três projetos que abordam questões de mobilidade: SIDAM – Sistemas de Informação Distribuída com Agentes Móveis (www.ime.usp.br/~sidam), que enfatiza aplicações distribuídas com mobilidade lógica (agentes móveis), usando como plataforma de desenvolvimento Objetos Distribuídos/CORBA; e SIAM – Sistemas de Informação com Agentes Móveis (www.dcc.ufmg.br/siam) que centraliza os estudos nas questões de gerenciamento de redes *ad-hoc*. Diferentemente, o **projeto ISAM** – Infra-estrutura de Suporte às Aplicações Móveis (www.inf.ufrgs.br/~isam), detalhado ao longo deste texto, objetiva fornecer uma infra-estrutura para a construção e execução de aplicações *pervasivas*. O projeto ISAM é parcialmente financiado pela FAPERGS (edital 003/2000) e engloba atividades desenvolvidas em quatro universidades: UFRGS, UFSM, UFPel e UNISINOS.

A motivação do projeto ISAM é derivada da observação de que poucas aplicações foram projetadas para experimentar com o cenário *pervasivo*. A inexistência de uma infra-estrutura para suportar este tipo de computação pode ser a razão da falta de aplicações. As aplicações existentes são desenvolvidas de forma *ad-hoc*, especifica para um problema ou domínio de aplicação (AUGUSTIN, 2001). Do ponto de vista de sistemas, a Computação *Pervasiva* apresenta desafios para a construção de aplicações que executam em um ambiente computacional altamente dinâmico, distribuído e heterogêneo (HENRICKSEN; INDULSKA; RAKOTONIRAINY, 2001). A aplicação deve ter a funcionalidade para o ambiente móvel e ainda embutir a solução de aspectos relativos ao sistema-base (já que este não existe). Isto as torna complexas, de difícil projeto e desenvolvimento, impede o reuso da solução e dificulta a evolução das aplicações.

A solução indicada no projeto ISAM é uma arquitetura de software organizada em camadas de vários níveis, abordados em subprojetos que tratam aspectos relativos ao suporte de linguagem e ferramentas de programação – ISAMadapt (descrita neste texto), e ao ambiente de execução – EXEHDA (YAMIN, 2004). Esta arquitetura supõe a existência de uma rede móvel de dimensão global, e baseia-se nas ferramentas fornecidas pela linguagem Java para sua implementação. Neste momento, o foco de pesquisa da arquitetura ISAM recai sobre o aspecto da adaptação ao contexto.

Integrando o projeto ISAM, esta proposta de tese aprofunda as questões relativas ao subprojeto ISAMadapt, responsável pela análise das abstrações necessárias a uma linguagem de programação para o cenário *pervasivo* e em disponibilizar as ferramentas necessárias para a programação das aplicações dentro deste cenário. Os próximos itens discorrem sobre a motivação, o problema e a proposta de solução do ISAMadapt.

1.2 Motivação do ISAMadapt: Aplicações no Ambiente *Pervasivo*

A pesquisa em Computação *Pervasiva*, que se acentuou muito a partir de 2001, visa atender um cenário de computação futura. Neste momento é um cenário hipotético tanto em termos de infra-estrutura física (hardware e rede) quanto de software/aplicações. É previsto que os futuros softwares façam mais tarefas para o usuário que o que foi esperado no passado. Existirão mais usuários, e conseqüentemente mais sistemas e interação entre eles, consumindo recursos (BANAVAR; BERNSTEIN, 2002).

Para ilustrar aplicações do cenário móvel atual, caracterizado como Computação Nômade, considere uma aplicação de agenda. A tecnologia hoje permite que se utilize a aplicação de agenda em vários dispositivos: telefones celulares, computadores de mão (*handheld*), *notebooks*. A inserção de um registro na agenda pode ser feita em qualquer dispositivo, porém esta informação somente estará disponível nos demais, se houver um processo de sincronização executado pelo usuário, uma vez que a implementação desta aplicação é única para cada plataforma. Para executar a sincronização, o usuário deve chamar a aplicação correspondente que sincroniza *handheld*-*notebook*, celular-*handheld*, celular-*notebook*. A questão que fica é: "por quê se têm diferentes implementações e dados para uma mesma aplicação em diferentes dispositivos, ao invés de uma única aplicação que se adapta ao dispositivo?". No ambiente *pervasivo*, a entrada de um registro é independente de dispositivo, pois os dados e o código da aplicação não são armazenados neles, mas estão espalhados, gerenciados transparentemente pelo sistema-base, e tornam-se disponíveis sob demanda. Nesta visão, os dispositivos são portais para as aplicações e dados, não repositórios de código e dados gerenciado pelos usuários (como hoje).

Um exemplo um pouco mais abrangente do cenário *pervasivo* é apresentado por Garlan, Steenskiste e Schemerl (2002) e resumido a seguir. Alissa é uma consultora de Tecnologia da Informação que visita muitos clientes, companhias, organizações profissionais e governamentais em todo o mundo. Freqüentemente viaja a trabalho. Para gerenciar seus numerosos contratos e sua agenda, faz uso de vários dispositivos. No escritório, usa um PC convencional. Em casa, e em viagens prefere o PDA. Na rua, o celular é também importante. No cliente, usualmente usa as facilidades computacionais para apresentações e relatórios. No hotel, usa serviços de informação Web. Mas em partes menos desenvolvidas do mundo, deve usar soluções locais. E através de todos esses dispositivos, plataformas computacionais e locais em que se encontra, Alissa deseja ter acesso às mesmas informações e funcionalidades. Além disso, muitas tarefas de gerenciamento pessoal, como agendamento de reuniões, são executadas

automaticamente e freqüentemente sem sua interação. Arranjos de viagens, reservas de hotel e vôos, baseados nas suas preferências, também não a envolvem diretamente (apesar de notificá-la dos resultados). Alissa costuma trabalhar mesmo quando em deslocamento. No seu escritório, lê seus e-mails utilizando o *desktop* da empresa. Quando chega o táxi para levá-la ao aeroporto, continua a atividade de e-mail, com seu PDA, durante o trajeto. Na sala VIP do aeroporto, enquanto espera, envia um arquivo contendo o relatório pedido por seu colega, que foi gerado enquanto estava no escritório.

Neste cenário, as atividades de Alissa são suportadas por um sistema que conhece suas preferências (modelo de usuário) e gerencia suas atividades (comportamento pró-ativo). Este sistema também dá suporte à *pervasividade* das aplicações e dados de Alissa (independência de plataforma/localização de dados e código), e ao chaveamento de contexto entre diferentes plataformas para continuar a execução da aplicação *e-mail* (semântica siga-me da aplicação).

Os componentes tecnológicos envolvidos neste exemplo são simples. Então, por quê parece um cenário futuro? A resposta talvez esteja no ditado “o todo é maior que a soma das partes”. O problema repousa na arquitetura e na engenharia no nível do sistema. A Computação *Pervasiva* visa libertar o usuário de estar associado a um dispositivo, plataforma computacional e rede nos quais armazena seus dados e configura suas aplicações. Nesta visão, o usuário poderá usar qualquer dispositivo disponível no lugar em que se encontra (próprio ou não) e terá acesso ao seu ambiente computacional independente de sua localização. A realização prática desta visão requer resolver muitos problemas de projeto e implementação (BANANAR; BERNSTEIN, 2002; HENRICKSEN; INDULSKA; RAKOTONIRAINY, 2001), e será uma fonte de pesquisa por muitos anos (SATYANARAYANAN, 2001).

1.3 O Problema

Ambientes complexos, como o *pervasivo*, exigem mudanças nos paradigmas de projetos: da orientação a objetos para a programação adaptativa ao contexto (*Context-aware Computing*). Software adaptativo ao contexto é aquele que usa as informações disponíveis sobre o ambiente para melhorar seu comportamento no tempo. A dificuldade de produzir tais softwares vem do fato de o projetista não pode prever todas as circunstâncias em que a aplicação poderá ser usada, e tomar todas as decisões em tempo de projeto. Isto deixa o software com vida curta, devido às constantes atualizações de situações não previstas. Este ambiente é dinâmico, único, com certo grau de incerteza, não-determinístico e tem influência do contexto em que está inserido. Nestes sistemas, o usuário deve sempre obter o que deseja (manter a funcionalidade), saber o que está acontecendo (ter controle sobre o ambiente), saber que não está sozinho – existem outros sistemas que cooperam, competem e se comunicam (fornecer consciência do contexto).

Portanto, o paradigma móvel/*pervasivo* impõe novos requisitos de operação aos sistemas. Entre eles:

a) inferir intenção do usuário;
b) gerenciar pró-ativamente as atividades do usuário;
c) mover estado de execução entre diversas plataformas e estados de contexto;
d) tornar dados e código disponíveis em qualquer lugar, em qualquer formato.

Atender a todos esses requisitos exige uma reformulação nos atuais sistemas, e nesta tese aprofunda-se a discussão sobre questões derivadas da alínea (c).

1.3.1 Primeira Abordagem do Problema: Transparência da Mobilidade

Para se iniciar a construir aplicações *pervasivas* poder-se-ia adotar a estratégia usada no projeto dos primeiros sistemas distribuídos, os quais estenderam as metodologias de programação seqüencial, baseadas em um único nodo, para distribuída, em vários nodos, e estender os sistemas distribuídos com mobilidade de código e gerenciamento da mobilidade física dos equipamentos. A seguir, faz-se uma análise dessa solução.

Quando desenvolvendo sistemas distribuídos, projetistas não tem de tratar explicitamente com problemas relativos à distribuição, como heterogeneidade, escalabilidade, compartilhamento de recursos, localização dos recursos, e outros. Os sistemas de suporte (*middlewares*, sistemas operacionais, linguagens de programação e ambiente de execução) fornecem ao programador abstrações de alto nível as quais lhe escondem a complexidade introduzida pelo gerenciamento da distribuição. Em geral, esses sistemas têm aderido à metáfora de caixa preta, onde a distribuição é escondida do usuário e do programador de forma a que o sistema pareça um bloco único, e os recursos remotos são mascarados como recursos locais. Esta **transparência** simplifica o desenvolvimento de aplicações e encoraja um estilo de programação no qual a inviabilidade de recursos ou a falha é vista como caso extremo, uma exceção. Porém, no ambiente de mobilidade global, onde usuários com seus dispositivos móveis deslocam-se continuamente, a indisponibilidade de alguns recursos é uma ocorrência freqüente e natural, não podendo ser considerada exceção.

Considerando o ambiente *pervasivo*, acredita-se que a mobilidade deve ser explorada pela aplicação (**consciência**), em vez de o sistema mascará-la, tornando-a transparente para a aplicação. Para tal, deve ser fornecido uma infra-estrutura de suporte a novos tipos de aplicações, conscientes da mobilidade, que permita ao programador tirar vantagem das características dinâmicas do ambiente, sem consumir muito de sua atenção.

Outro problema em usar plataformas distribuídas para construir softwares para o ambiente móvel é que estas foram construídas com **suposições fixas sobre o ambiente** no qual executam. As aplicações distribuídas são compostas através de interfaces dos componentes do sistema de suporte, como objetos distribuídos, as quais simplificam o desenvolvimento. Esta metodologia traz um **alto acoplamento** entre os componentes da aplicação, uma vez que eles diretamente invocam outros componentes através de suas interfaces. Como resultado, é difícil adicionar novo comportamento à aplicação. Estender um componente requer alterar sua interface e esta é limitada pelo grau de extensão projetado no sistema.

Nos sistemas distribuídos, os nodos são estacionários, os serviços e recursos disponíveis são conhecidos. A mobilidade do usuário traz como conseqüência um ambiente de execução dinâmico, que se altera conforme o usuário/nodo se desloca. Logo, os requisitos para tratar com a mobilidade se opõem aos pressupostos com os quais os sistemas distribuídos tradicionais são construídos.

Por exemplo, a mobilidade, juntamente com a portabilidade e a conectividade no meio sem fio, afetam o mecanismo de interação. Em sistemas distribuídos, em geral, o suporte à **comunicação** é baseado em **mecanismos síncronos ponto-a-ponto**. Os nodos participantes da comunicação são conhecidos, estacionários e podem ser nomeados. A rede é estável, com alta largura de banda, e localização fixa dos nodos; os serviços e recursos são conhecidos, com localização fixa dos servidores. A conexão pode ser garantida enquanto dura a interação. No entanto, o ambiente móvel naturalmente impõe restrições que contrastam com este ambiente: o nodo móvel pode não estar conectado (por questões de economia de energia, este opera normalmente desconectado da rede), seu endereço de rede não é fixo; a conexão com a rede é intermitente, a largura de banda

é baixa; o endereço dos servidores pode não ser conhecido a priori; ao enviar uma mensagem para um nodo, este pode ter trocado de endereço devido ao deslocamento do usuário, ou estar desligado; a conexão não é garantida durante toda a interação. Desta forma, o ambiente *pervasivo* requer um mecanismo de **comunicação assíncrona**, nodos que se comunicam não necessitam estar conectados para o processo de comunicação, e **anônima,** nodos não precisam ser nomeados para a comunicação.

Em geral, os sistemas distribuídos **encapsulam dados e funcionalidade** dentro de uma única abstração, como os baseados em objetos. Isto limita como os dados podem ser usados e restringe o compartilhamento, pesquisa e filtragem de dados.

Por outro lado, adotar a solução WWW (*world wide web*), que não é baseada em interface e não encapsula dados e funcionalidade, também apresenta limitações. Primeiro, esta requer **operações conectadas, síncronas** entre cliente e servidor. Segundo, esta coloca no **usuário a decisão de adaptação/reação** em caso de indisponibilidade de recurso, como o servidor. Terceiro, esta parece **não acomodar tecnologias úteis** para a construção de aplicações *pervasivas,* com mobilidade física e descoberta de serviços.

Conclui-se que os sistemas existentes de suporte à distribuição não são capazes de tratar adequadamente a mobilidade, principalmente as necessidades impostas pela mobilidade física de usuário e nodos (*pervasiva*).

Esses problemas têm limitado o uso atual do conceito de mobilidade para aplicações com mobilidade lógica (dados/código), implementadas com o conceito de agentes móveis – componentes de software autônomos que migram para sítios, com um trajeto pré-determinado, para realizar uma tarefa (BELLAVISTA; CORRADI; STEFANELLI, 2001). O conceito de mobilidade lógica – do estado da execução - foi proposto como uma solução para implementação de aplicações móveis (CARDELLI, 1998). Segundo este conceito, a execução da aplicação não precisa mais se restringir a um único nodo da rede, podendo migrar por vários nodos de acordo com as necessidades da tarefa que se propõe a realizar. Concorda-se com o argumento de que esta solução contribui para diminuir a dependência das aplicações em relação à rede, e para permitir a operação em modo desconectado. Porém, somente a migração para locais escolhidos não é suficiente para atender os requisitos das aplicações com mobilidade *pervasiva*.

1.3.2 Segunda Abordagem do Problema: Consciência da Mobilidade e do Ambiente

As considerações anteriores levantam a questão de “como construir aplicações móveis para o ambiente *pervasivo* usando a tecnologia disponível”. Vê-se que, enquanto aplicações móveis compartilham as mesmas dificuldades e desafios encontrados em aplicações distribuídas (sobre redes convencionais), a natureza de seu ambiente impõe novas dificuldades não encontradas em outro lugar (HENRICKSEN; INDULSKA; RAKOTONIRAINY, 2001).

Existem três propriedades fundamentais para as aplicações *pervasivas*: redes sem fio (conectividade), habilidade de trocar de localização (mobilidade), e habilidade de usar dispositivos portáteis (portabilidade) (AUGUSTIN, 2002b). Essas propriedades impõem restrições que são endêmicas aos sistemas móveis. O ambiente *pervasivo* altera-se dinamicamente. Condições de conectividade variam da total desconexão à completa conectividade. Os recursos disponíveis para os componentes móveis não são estáticos. Além disso, a localização do elemento móvel altera-se assim como a configuração da rede e o centro da atividade computacional. Assim, o sistema móvel é apresentado com recursos que variam em número e qualidade.

A questão, mais refinada, que se apresenta é “como tratar dessas restrições e estruturar as aplicações móveis de forma a permitir a execução adequada destas”. A oferta variável de recursos, tanto quanto a diversificada demanda destes, sugere que as aplicações devem adaptar-se a essas variações. Sendo assim, a adaptação é o mais importante requisito para as aplicações móveis atingirem o grau de desempenho que atenda as necessidades dos usuários.

Argumenta-se que uma aplicação móvel deve ser projetada com base em várias funcionalidades que se ajustam ao ambiente corrente – contexto - e que são selecionadas automaticamente pelo sistema de gerenciamento das aplicações. Essa nova visão de aplicações, que modificam seu comportamento em função das alterações no seu contexto de execução, exige novas abstrações nas linguagens de programação e um sistema de suporte integrado à linguagem.

Além disso, observou-se uma falta de suporte genérico à construção de aplicações móveis. A grande maioria das aplicações móveis com comportamento adaptativo é específica, suas soluções são limitadas a um problema ou domínio de aplicação, como multimídia (NOBLE, 2000). A adaptação, em geral, se refere à reação a alteração de um recurso que é monitorado pelo sistema, como largura de banda ou latência. No entanto, as aplicações móveis se valem de vários outros recursos os quais também sofrem variações que interferem na execução da aplicação, tais como energia (bateria) e localização (que por sua vez, permite ou não o acesso a outros recursos). Além disso, essas adaptações se baseiam em extensões de soluções de aplicações distribuídas, considerando, em geral, somente um mecanismo de adaptação/reação, como migração (RANGANATHAN; ACHARYA; SALTZ, 1997) ou replicação (BAGGIO, 1999).

No sistema *pervasivo*, o contexto de execução das tarefas do usuário e, conseqüentemente, contexto do sistema, altera-se com o tempo conforme o usuário se desloca. Alteração de contexto é desafiante na arquitetura do sistema de duas formas: (a) definir mecanismos e padrões que suportem adaptação ao contexto corrente; (b) definir metodologias e modelos que facilitem o projeto de tais sistemas.

Aplicações móveis conscientes do contexto são difíceis de projetar e implementar devido à necessidade de um comportamento dinamicamente adaptativo às condições ambientais em que executam (DEY, 2000). Os modelos de software atuais não acomodam a generalidade necessária para a produção dessa classe de aplicações. No campo da mobilidade, acredita-se que os modelos de software assumem um nível de significado crescente na implementação de aplicações. Estes podem derivar de *middlewares* e ferramentas associadas (*frameworks*, APIs) e são significativamente diferentes entre as propostas analisadas (ver Capítulo 7). A falta de unidade dificulta a simplicidade e sua utilização efetiva no projeto de aplicações móveis. Alguns trabalhos têm fornecido *frameworks*, que dão suporte às fases da adaptação: monitoração e notificação (DEY, 2000; WELLING; BADRINATH, 1998) e reação (EFSTRATIOU et al., 2000). Porém, a generalidade fornecida é parcial, pois objetivam um domínio específico de aplicações. Além disso, a mobilidade física não é focalizada nesses trabalhos, o que dificulta sua utilização no domínio da Computação *Pervasiva*.

1.3.3 O Problema Específico da Tese

Nessas pesquisas (ver Capítulo 7) observa-se a falta de suporte tanto para o projeto quanto para a programação de aplicações móveis conscientes do contexto. Esta falta se deve a não existência de abstrações genéricas para a consciência de contexto baseadas em noções de reusabilidade e flexibilidade para adequar-se a especialidades de cada aplicação, e a falta de um suporte à execução automática da adaptação expressa pela aplicação.

Pode-se dizer que a existência de aplicações móveis no ambiente *pervasivo* é reduzida devido a fatores como: (i) inexistência da infra-estrutura de rede móvel de amplitude global; (ii) inexistência da infra-estrutura de suporte à execução; (iii) inexistência de suporte à produção de software móvel que simplifique e agilize este processo, naturalmente mais complexo que o do ambiente distribuído. O primeiro fator está fora do escopo deste trabalho e, portanto, não é abordado. Os dois outros fatores são objetos de estudo do projeto ISAM.

Acredita-se que a utilidade das aplicações móveis é ampla, e aberta a novos domínios, e também aberta a domínios dominados hoje pela computação fixa. Vê-se um interesse crescente na indústria para a disponibilização da infra-estrutura de rede móvel em escala global. Logo, este ambiente exigirá aplicações. Assim, um movimento em direção a **soluções de propósito geral** é necessário, o qual permita disseminar a produção de aplicações para o ambiente móvel. Estas podem ser alcançadas com construções específicas de linguagens, que encapsulem a complexidade e a interação com a plataforma de suporte, e oferecem uma semântica bem definida e uma interface de programação simples.

Não é do conhecimneto da autora nenhuma **linguagem de programação** com características adequadas ao ambiente *pervasivo*. Em geral, os sistemas de código móvel adotam a linguagem Java pela facilidade de mobilidade de código, que permitem sobrepor a limitação do particionamento estático da aplicação orientada a objetos. Porém, o uso simples deste mecanismo é restritivo para muitas aplicações com mobilidade física, pois não tratam de questões relativas a este campo, como desconexões e facilidades para adaptação consciente do contexto. Linguagens de agentes móveis também falham pela falta de serviços baseados no estado do ambiente de execução, e na falta de uma infra-estrutura para gerenciamento de aplicações em escala global.

Analisando o problema sob a ótica dos sistemas adaptativos, vê-se que os sistemas atuais também são limitados, pois abordam o problema da adaptação em dois extremos: (a) enfatizam a transparência da solução, onde o sistema é o responsável pela adaptação; (b) soluções específicas para a aplicação, que são difíceis de projetar (o programador deve tratar muitas questões) e de reusar.

Concluindo, soluções para a construção de sistemas móveis têm se baseado na alteração de sistemas distribuídos pré-existentes, projetados para o ambiente fixo que tem como premissas a conexão permanente e disponível, os recursos estacionários e sem limitação de cotas. Além disso, em sua maioria, as soluções para o projeto de sistemas móveis abordam somente um ou outro aspecto da mobilidade. Porém, aplicações móveis reais são difíceis de serem projetadas, considerando somente o aspecto de migração de estado, por exemplo. Ou considerando apenas informações de contexto, sendo que o programador deve fazer todo o tratamento de reconfiguração/adaptação de comportamentos alternativos.

O projeto ISAM argumenta que uma alternativa viável para construir aplicações *pervasivas* é construí-las dentro de uma arquitetura de sistema que forneça um conjunto de serviços básicos e integrados, projetados especificamente para este ambiente. Esses serviços incluem: distribuição automática e instalação de aplicações, gerenciamento distribuídos dos componentes da aplicação, gerenciamento do contexto de execução das aplicações, gerenciamento do comportamento adaptativo da aplicação e do próprio sistema de execução. Nesta perspectiva, o objetivo do subprojeto ISAMadapt é de fornecer ferramentas que permitam programar e experimentar com aplicações que tem novos requisitos derivados da *pervasividade*, e identificar a influência desta no comportamento da aplicação e do sistema de gerenciamento. As ferramentas integrantes

do ISAMadapt procuram facilitar[2] a construção de aplicações móveis conscientes do contexto e são implementadas com base nos serviços fornecidos pelo ambiente de execução EXEHDA.

1.4 A Tese

A imprevisibilidade do ambiente *pervasivo* determina que aplicações não devem ser desenvolvidas de maneira *ad-hoc*, considerando cada condição específica. A aplicação deve ser construída para ser adaptável ao contexto. Desta forma, é necessário fornecer uma infra-estrutura para fornecer consciência do contexto e gerenciar a adaptação, objetivando alcançar uma generalidade que permita flexibilidade e expressividade para um grande número de aplicações *pervasivas* com comportamento consciente do contexto.

A flexibilidade é alcançada com (i) a separação de conceitos – diferentes aspectos do sistema são isolados em diferentes componentes que localizam o efeito da adaptação ao contexto; (ii) mecanismos projetados para adaptação através de várias formas parametrizadas.

Sob a perspectiva do desenvolvimento do software, tem-se a hipótese de que "a aplicação é construída e evolui mais facilmente se **abstrações de alto nível** e **ferramentas** que as implementam são fornecidas". As abstrações devem conduzir a uma abordagem mais declarativa que procedimental. O projetista deve se concentrar em definir as situações ambientais – contexto - em que a aplicação pode executar, e mais precisamente, que informações do ambiente são relevantes e como reagir a elas, do que saber como estas são adquiridas ou gerenciadas. Desta forma, as abstrações se bem projetadas podem permitir a implementação da noção de **separação de conceitos**, onde a expressão da funcionalidade da aplicação permanece a mesma independentemente da expressão do processo de adaptação.

O que nos conduz à tese de que: *identificando e suportando um conjunto de abstrações, ferramentas e serviços para tratar com a adaptação ao contexto, disponibilizados em uma linguagem de programação, torna-se mais fácil projetar, construir e evoluir aplicações móveis que executam em um ambiente pervasivo.*

Refinando esta tese, tem-se que: *para simplificar a construção de aplicações móveis com comportamento consciente do contexto é necessário que a* ***linguagem de programação*** *ofereça abstrações que permitam expressar o comportamento adaptativo ao contexto desejado pela aplicação, bem como a* ***colaboração do sistema de suporte*** *(middleware, toolkit e runtime) na execução da aplicação.*

A adaptação dos componentes da aplicação deve ser definida em termos de adaptações de componentes individuais e políticas de colaboração com o sistema de gerenciamento e execução da adaptação. A adaptabilidade deve ser seletiva.

Considera-se que a completa automação da adaptação não é alcançável devido à natureza da adaptação estar semanticamente ligada a da aplicação em particular. Os mecanismos que a aplicação usa para se adaptar são certamente específicos, porém as decisões sobre quando e como a (re)configuração ocorre são mais apropriadamente realizadas de um modo centralizado pelo sistema de suporte, que detém o controle do

[2] Salienta-se que o objetivo não é reduzir a natural complexidade dos sistemas móveis adaptativos, somente fornecer ferramentas que simplifiquem a sua construção.

todo. O sistema, equipado com informações sobre seu estado corrente e indicações de quais componentes das aplicações permitem a (re)configuração, pode adaptar uma ou todas aplicações a fim de aumentar a utilização de recursos do sistema. A vantagem é que o sistema pode ajustar-se não só a uma aplicação individual, mas a um conjunto de aplicações. A Figura 1.1 ilustra o relacionamento entre a linguagem e o sistema de suporte à execução no ambiente *pervasivo*.

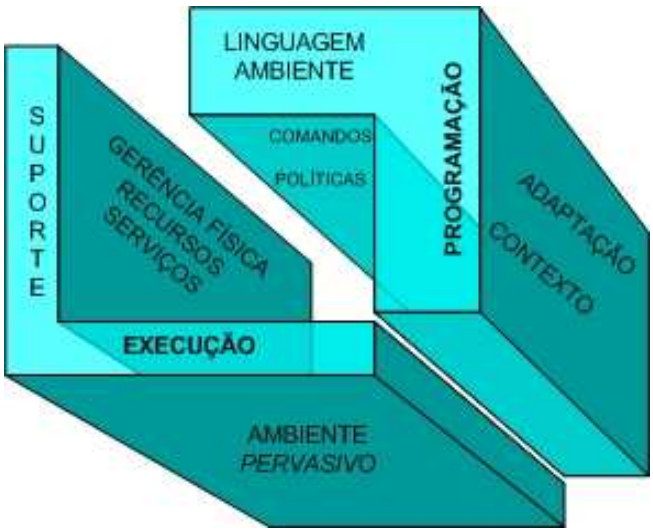


Figura 1.1: Relacionamento entre Linguagem e Sistema de Suporte à *Pervasividade*

1.5 Tópicos de Pesquisa

Motivado pelo estado-da-arte no campo da Computação Móvel em direção à Computação *Pervasiva*, os tópicos de pesquisa relativos a esta tese podem ser agrupados na questão: "*como projetar e implementar uma aplicação móvel que se adapta às variações ambientais – contexto – e executa em um ambiente pervasivo?*". Com esta questão em mente, outras subquestões são identificadas na direção de obter uma resposta:

(a) quais as características e necessidades destas aplicações?
(b) como ocorre o processo de adaptação ao contexto?
(c) qual o suporte necessário para o processo de adaptação?
(d) qual o modelo de programação e interação necessário?
(e) como codificar aplicações conscientes do contexto?

Essas questões são respondidas ao longo deste texto.

Nesta tese, o escopo é a adaptabilidade ao contexto e como esta pode ser implementada em uma linguagem de programação. O ambiente de execução que dá suporte à execução da aplicação e o suporte de rede móvel são assumidos como existentes, e não serão detalhados. Um estudo relativo aos aspectos do ambiente de execução e de rede está sendo realizado pelo subprojeto EXEHDA (E*nvironment Execution for High Distributed Applications*), outro integrante do projeto ISAM (YAMIN, 2004).

1.6 Estrutura do Texto

Este texto está organizado em tópicos distribuídos em nove Capítulos e dois Apêndices.

No segundo Capítulo, apresentam-se os principais conceitos relativos à visão de mobilidade no projeto ISAM, a taxonomia de aplicações móveis com comportamento adaptativo resultante dos estudos comparativos das aplicações móveis encontradas na literatura, e os requisitos que embasam a arquitetura de software ISAM. Com o objetivo de esclarecer a nomenclatura usada neste texto, o Apêndice A apresenta uma organização de conceitos sobre Computação Móvel, os cenários e as tecnologias disponíveis para produção de aplicações móveis.

O terceiro Capítulo é dedicado à arquitetura de software ISAM, discutindo aspectos relativos a esta tese. O quarto Capítulo aborda o subprojeto ISAMadapt que objetiva identificar as abstrações necessárias para expressar o comportamento adaptativo ao contexto no nível de uma linguagem de programação e disponibilizar ferramentas que as implementam. O Apêndice B discute as questões de projeto do Serviço de Reconhecimento de Contexto, `ISAMcontextService`, relacionado ao ISAMadapt. O quinto Capítulo registra as questões relativas à implementação dos componentes do ISAMadapt. O sexto Capítulo apresenta um estudo de caso, e exemplos de aplicações que podem ser projetadas com o ISAMadapt. No sétimo Capítulo são comparados com o ISAMadapt os trabalhos relativos aos diversos aspectos presentes na solução: sistemas de Computação Móvel, ambientes de Computação *Pervasiva*, paradigmas para a programação adaptativa, linguagens para adaptação. O nono Capítulo registra as conclusões, os problemas não tratados, as contribuições, as publicações, e os temas de trabalhos futuros relativos a essa tese.

2 CONSEQÜÊNCIA DA MOBILIDADE NOS SISTEMAS DE COMPUTAÇÃO

A pesquisa bibliográfica realizada sobre a Computação Móvel concluiu que este conceito tem significado diferente para diferentes pessoas. Isto se deve ao fato de a área ser muito recente, seus conceitos e definições não estão claramente definidos ou delimitados, e em constante modificação. Devido ao grande apelo comercial, às vezes, os conceitos são usados de forma incorreta. Alguns problemas de conceituação são também decorrentes de traduções, como a tradução de *Mobile Computing* e *Mobile Computation* para Computação Móvel, e *portability* para referir-se à propriedade de um componente ser portável (traduzido para outra plataforma, portabilidade de software) ou portátil (transportável, portabilidade de hardware). Esta situação tem levado a uma confusão de nomenclatura. Logo, sente-se necessidade de definir os conceitos que norteiam este trabalho. Como não existem ainda definições amplamente aceitas, procurou-se adotar aquelas que mais se assemelham à visão da autora sobre a área de Computação Móvel. O Apêndice A aborda essa temática. Neste Capítulo abordam-se questões relativas ao impacto da mobilidade física nos sistemas de computação, e enfatiza-se o surgimento de novos tipos de aplicações e a taxonomia proposta.

2.1 A Linha de Evolução da Computação Pervasiva

Na evolução em direção à Computação *Pervasiva*, duas tecnologias a antecedem: a computação distribuída e a computação móvel. A computação distribuída foi a primeira etapa para o ambiente *pervasivo*, através da introdução do acesso remoto a recursos de informação e comunicação. A Web pode ser considerada a primeira rede *pervasiva*, e é uma escolha atraente para experimentar com o conceito de distribuição. Esta também contribui com o crescimento da cultura e inserção da computação na vida diária dos usuários. A computação móvel, por sua vez, vem da utilização de dispositivos portáteis (PDAs) e celulares com acesso à rede via meio sem fio. O Apêndice A detalha essa evolução, na visão da autora.

O objetivo da computação móvel de "a qualquer tempo em qualquer lugar" é essencialmente uma abordagem reativa para o acesso à informação, o qual prepara o caminho para o objetivo pró-ativo da computação *pervasiva* de "todo tempo em qualquer lugar". A Figura 2.1 mostra a visão corrente de que a Computação *Pervasiva* é um superconjunto da Computação Móvel[3] (SAHA; MUKHERJEE, 2003). Em adição

[3] Computação Móvel refere-se à tecnologia atual em mobilidade, enquanto que Computação *Pervasiva* refere-se à tecnologia no futuro próximo.

à mobilidade, a *pervasividade* requer suporte para interoperabilidade, escalabilidade, consciência e invisibilidade.

2.2 Diferença entre Sistema Distribuído e Sistema Móvel

Como a mobilidade do usuário/nodo interfere na computação não é ainda uma questão bem entendida, uma vez que existem poucas aplicações desenvolvidas para este ambiente. Desta forma, um dos primeiros objetivos deste trabalho foi procurar entender a influência da mobilidade nas aplicações distribuídas, ou mais precisamente, qual a diferença entre aplicações distribuídas e aplicações móveis.

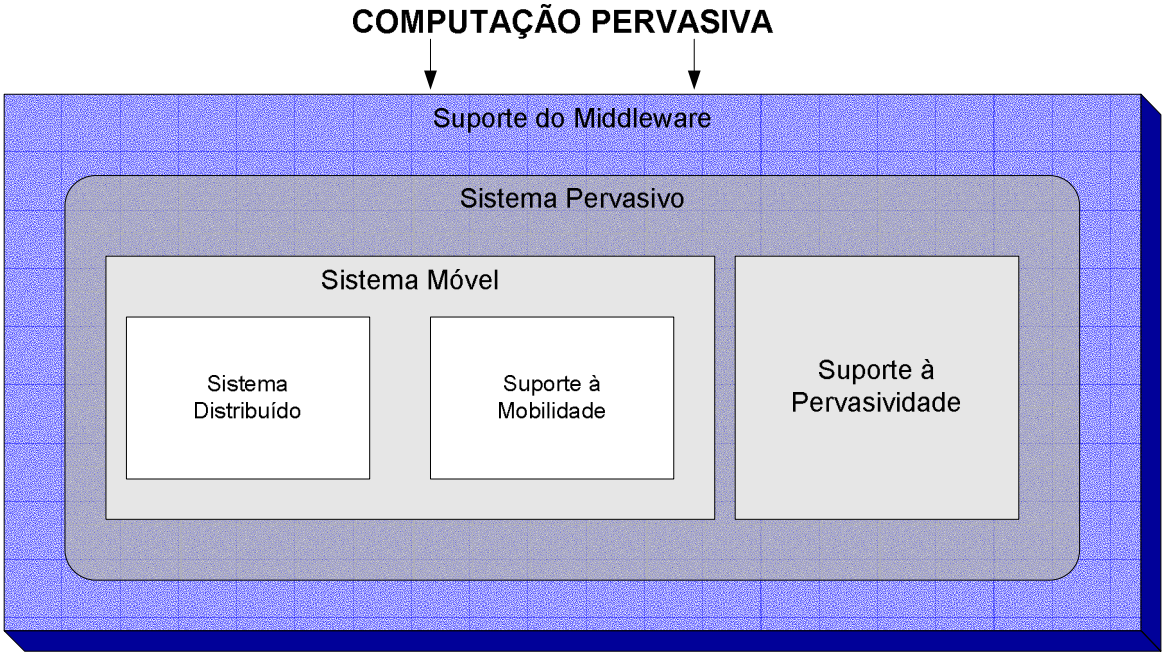


Figura 2.1: O Sistema *Pervasivo*

Uma aplicação distribuída consiste em uma coleção de componentes, distribuídos sobre vários computadores (nodos) conectados via uma rede de computadores. Esses componentes interagem uns com os outros para trocarem informações ou acessarem serviços. Esta definição se aplica a ambos: sistema distribuído fixo ou móvel. A diferença está subjacente ao conceito e deriva de três aspectos:

a) tipo de nodo. Nos sistemas distribuídos móveis, os nodos têm a propriedade de portabilidade e mobilidade[4], diferente dos nodos fixos dos sistemas distribuídos tradicionais;
b) tipo de conexão de rede. Usualmente, os nodos fixos estão permanentemente conectados à rede através de *links* de alta largura de banda. Por outro lado, nodos móveis operam em modo desconectado na maior parte do tempo. Isto se deve a dois fatores: (*i*) para economia de energia, que provém da bateria que em geral tem poucas horas de duração (IMIELINSKI; VISWANATHAN; BADRINATH, 1997).

[4] Considerando uma abordagem purista, a propriedade de mobilidade pertence ao usuário portando o equipamento, e não ao equipamento em si. Neste texto, optou-se por seguir o termo usado na literatura: *mobile device*.

Desta forma, diz-se que a conexão no ambiente móvel é *intermitente*, e a desconexão é *planejada*; (*ii*) o meio sem fio é altamente propenso a interferências ambientais que causam a interrupção abrupta da conexão.

c) tipo de contexto de execução. O contexto, que inclui recursos internos e externos à aplicação, como tamanho da tela, memória disponível, largura de banda, localização, pode influenciar seu comportamento. Em sistemas tradicionais, este ambiente é mais ou menos estático: banda é alta e contínua, localização dos nodos é conhecida, nodos podem ser adicionais ou removidos – mas isto acontece com pouca freqüência. Em oposição, ambiente móvel é extremamente dinâmico.

Para os pesquisadores, muitas das questões envolvendo sistemas móveis são as mesmas dos sistemas distribuídos, mas as soluções são diferentes. O relacionamento entre sistemas distribuídos e sistemas móveis foi discutido *no IEEE I Workshop on Mobile Computing Systems and Application*, CA, USA, dezembro de 1994, e a diferença mais significativa encontrada foi o objetivo de "transparência da localização" nas aplicações distribuídas e de "consciência da localização" nas aplicações móveis, e as conseqüências que derivam deste objetivo.

2.2.1 Conseqüências da Mobilidade do Usuário

Como visto, os sistemas móveis diferem dos sistemas distribuídos, pois não existe uma infra-estrutura fixa. Além disso, requisitos não funcionais existentes nos sistemas distribuídos tradicionais, como escalabilidade e segurança, adquirem uma dimensão maior devido à mobilidade e à portabilidade. Computadores móveis podem entrar e sair dos domínios de uma rede de forma não-determinística, conforme o usuário se desloca. O número potencial de nodos no sistema também é maior, e os problemas de segurança no meio sem fio são mais críticos que os atuais. Os aspectos de tolerância a falhas parecem agora serem mais severos. Enquanto que falhas ocorrem eventualmente em sistemas distribuídos, e são consideradas exceções, nos sistemas móveis estas são inerentes a sua natureza, e não podem ser tratadas com o conceito de exceções.

Em termos de organização do software, sistemas distribuídos apresentam a dimensão temporal e espacial, enquanto que sistemas móveis acrescentam outras duas dimensões, pessoal e social, às aplicações.

As conseqüências da mobilidade do usuário levantam implicações, particularmente em termos de propriedades dos sistemas convencionais. As propriedades mais afetadas pela mobilidade são: conectividade, disponibilidade de recursos, localização de recursos, custo e consumo de energia. Devido ao fato de os clientes móveis serem compactos e leves, estes, tipicamente, são pobres em recursos (relativamente ao ambiente *desktop*). A conectividade à rede via meio sem fio sobre uma grande área tende a alta variabilidade na largura de banda[5], na latência, na confiabilidade e no custo. Além disso, o conjunto de serviços disponíveis em cada localização pode diferir. Gerenciamento da energia freqüentemente exige que as ações sejam adiadas, evitadas ou executadas mais lentamente para prolongar a vida da bateria. O custo relativo de acesso aos serviços e recursos varia conforme o movimento do cliente (localização e padrão de deslocamento). Além disso, a natureza da mobilidade e da portabilidade tem um impacto negativo na robustez e na segurança.

[5] Células de diferentes tamanhos oferecem variação na banda: alta largura de banda com baixa razão de erros em picocélulas, baixa banda passante e alta razão de erros em macrocélulas (KUNZ; BALCK, 1999).

Concluindo, a mobilidade potencializa os problemas ainda não solucionados na computação distribuída, principalmente considerando ambientes altamente distribuídos, como a computação em grade (*Grid Computing*).

Por conseguinte, a produção de software no ambiente móvel é complexa, seus componentes são variáveis no tempo e no espaço. Produzir aplicações para o ambiente móvel é bastante desafiador (HENRICKSEN; INDULSKA; RAKOTONIRAINY, 2001), porque, tipicamente, executam em ambientes heterogêneos e exibem um alto grau de compartilhamento de recursos, o qual resulta numa contínua alteração do ambiente (desconexões freqüentes, variação na banda passante, redes heterogêneas, riscos de segurança). Isto é particularmente verdadeiro quando redes de diferentes tecnologias com diferentes características de desempenho estão envolvidas. O meio no qual as operações de comunicação se processam pode variar de milisegundos a segundos dependendo do ambiente corrente e das condições da rede (STEENKISTE, 1999). O desafio que se apresenta, para as aplicações, é que o nível de serviço e a disponibilidade de recursos esperados são imprevisíveis.

2.2.2 Adaptação e sua Necessidade no Ambiente Móvel

Pesquisadores concordam que esses sistemas, para serem efetivos e apresentarem um desempenho compatível com a expectativa do usuário, exigem a **capacidade de adaptação** às freqüentes e rápidas alterações no ambiente de execução durante o curso de execução da aplicação (KATZ, 1994; NOBLE, 2000; SATYANARAYANAN, 2001; HENRICKSEN; INDULSKA; RAKOTONIRAINY, 2001). Para tal, é entendido que sistemas devem ter consciência da localização e da situação, e deve tirar vantagem desta informação para dinamicamente configurar-se. Além disso, as aplicações podem fornecer *feedback*, habilitando a expectativa do usuário em, implicitamente, alterar seu modo de trabalho, fazendo melhor uso dos recursos disponíveis.

Um sistema é adaptável se é capaz de ser adaptado ou adaptar-se. Adaptação consiste em modificações ou ajustes para preencher novos usos ou novas condições. A noção de **adaptação** no ambiente móvel é ampla e requer uma definição. Observa-se que a adaptação nos sistemas móveis pode se referir a diversas noções, dependendo dos objetivos e domínio das aplicações. Em muitos sistemas, a adaptação diz respeito ao uso de técnicas de redução/transformação/*caching* dos dados para trafegarem na rede, sendo este processo automaticamente disparado pela alteração na largura da banda (AUGUSTIN, 2001). Várias outras formas de consciência da mobilidade têm sido propostas, e estão descritas no Capítulo 7.

Assim, enquanto em sistemas distribuídos a aplicação pode ser adaptável, em sistemas *pervasivos* ela é adaptável. Porém, há pouco consenso na comunidade científica sobre como especificar e gerenciar a adaptação. Cada autor aplica adaptabilidade em diferentes níveis de sua arquitetura: modelando explicitamente políticas de adaptação (CORRADI et al., 2001), gerenciando adaptação no nível de aplicação (ANDRÈ; SEGARRA, 2000a) ou alcançando adaptação de forma transparente na infra-estrutura base (JOSEPH; TAUBER; KAASHOEK, 1996; RANGANATHAN; ACHARYA, SALTZ, 1997).

Embora abordem aspectos importantes das aplicações móveis, estes falham em fornecer um modelo de adaptação genérico, capaz de dar suporte às aplicações móveis de propósito-geral. Esta questão é tratada nesta tese, a qual focaliza as abstrações para expressar um comportamento adaptativo no nível de linguagem de programação. Além disso, salienta-se que nenhum desses sistemas aborda o ambiente de Computação *Pervasiva*.

2.3 Taxonomia das Aplicações Móveis com Comportamento Adaptativo

Para melhor entender o conceito de adaptação, e sua utilização nas aplicações móveis, foram analisados vários trabalhos (ver Capítulo 7). O resultado desse estudo é uma taxonomia que procura identificar as características e necessidades das aplicações móveis com comportamento adaptativo (AUGUSTIN, 2002b), apresentadas a seguir. A análise dos sistemas móveis nos auxiliou a entender o processo de adaptação e a selecionar as abordagens apropriadas para diferentes domínios de aplicação. Como a intenção da arquitetura ISAM é ser de propósito-geral, as escolhas foram feitas com essa premissa.

2.3.1 Dimensões e Níveis de Adaptação

A adaptação nas aplicações móveis pode ocorrer basicamente em quatro dimensões: temporal, espacial, pessoal e social (AUGUSTIN, 2002a). A Figura 2.2 mostra as perspectivas de adaptação no ambiente móvel, considerando as entidades responsáveis por ela (controle): usuário, aplicação, sistema de suporte e as dimensões possíveis.

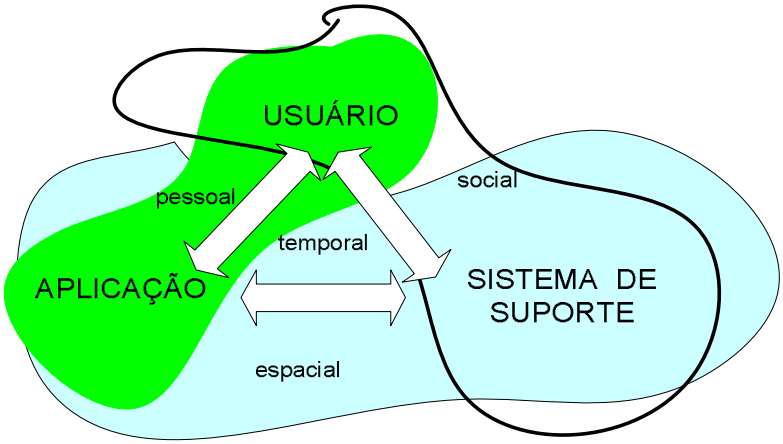


Figura 2.2: Perspectivas do Gerenciamento da Adaptação na Mobilidade

Adaptação **temporal** refere-se a quando ocorre a adaptação, em tempo de carga (semi-estática) e/ou em tempo de execução (dinâmica). Adaptação **espacial** é relativa a quem sofre a adaptação, e que elementos computacionais são essenciais para a aplicação e influenciam seu comportamento. Esta pode incluir entidades, atividades e localização. Diferente de outros sistemas adaptativos, a mobilidade introduz a dimensão **pessoal**: relativa ao padrão de comportamento e preferências do usuário móvel. A dimensão **social** aparece quando o ambiente, composto por várias aplicações, influencia o comportamento da aplicação executando no dispositivo móvel.

Além disso, adaptação ocorre em níveis relativos ao ponto na arquitetura de software onde esta é realizada: na plataforma fundamental (rede e sistema operacional), no sistema de suporte (*middleware, runtime*) ou na aplicação.

Muitas combinações entre essas dimensões e níveis de adaptação são possíveis. Observa-se que as aplicações podem também diferir no foco da adaptação: recurso, contexto ou situação. Estas observações resultaram na taxonomia proposta para as aplicações móveis com comportamento adaptativo (AUGUSTIN 2002b).

2.3.2 Taxonomia Proposta

Adaptação pode ser introduzida em todos os níveis de uma aplicação, e pode tomar diferentes formas, de protocolos adaptativos a interfaces de usuário adaptativas e qualidade de serviço adaptativa. A experiência com o projeto ISAM permitiu identificar algumas classes de aplicações e os aspectos que são importantes para elas. Assim, propôs-se uma primeira taxonomia para expressar o comportamento adaptativo das aplicações móveis (AUGUSTIN, 2002b).

Adaptação requer diferentes perfis de utilização dos elementos computacionais. Como conseqüência, estão aparecendo diversos tipos de aplicações nas quais o comportamento é determinado pela sensibilidade a alguma entidade do ambiente. O ambiente é definido por elementos computacionais que podem ser mensurados, tais como largura de banda, latência, consumo de energia, localização e preferências do usuário.

A taxonomia para aplicações móveis adaptativas é mostrada na Figura 2.3. A base da classificação é a resposta a três questões: “quem controla a adaptação – aplicação ou *middleware*?”, “que elemento computacional dispara a adaptação – recurso, localização ou contexto?”, “como é feita a adaptação – paramétrica, conteúdo ou funcional?”.

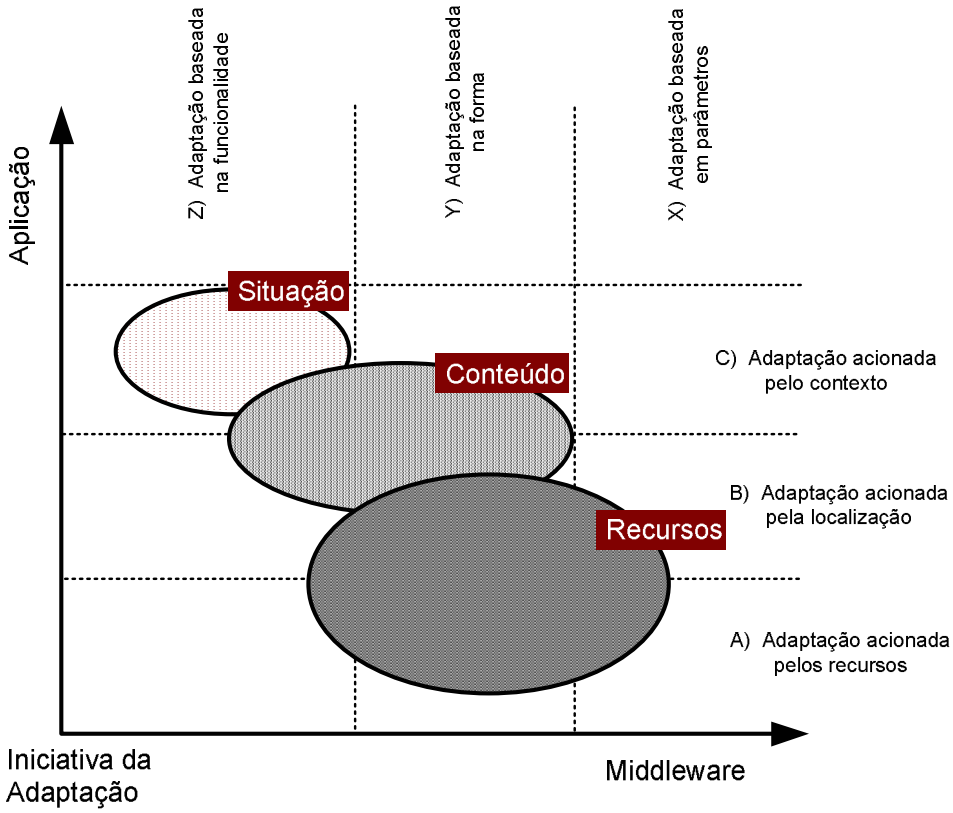


Figura 2.3: Taxonomia de Aplicações Móveis Adaptativas. A elipse representa os tipos de aplicações. As linhas horizontais (A,B,C) representam a sensibilidade ao elemento computacional, enquanto que as linhas verticais (X,Y,Z) representam as estratégias de adaptação, dispostas em ordem de complexidade.

A responsabilidade do processo de adaptação move-se entre dois limites: o sistema (automático/transparente) e a aplicação (programação/explícito). Para identificar o domínio das aplicações que, por um lado, tiram vantagem da adaptação automática e, por outro, devem ter interferência da aplicação ou usuário, separaram-se os elementos computacionais cujas trocas em seu estado disparam o processo de adaptação: recurso,

localização e contexto. Recursos incluem entidades físicas e lógicas dentro de uma faixa local. Exemplos são relativos à rede, ao dispositivo móvel e estacionário, aos arquivos, aos programas e outros. Localização é a posição física do usuário com seu equipamento portátil. A localização é o conceito central no ambiente móvel, pois determina o contexto da aplicação. Contexto é um conceito amplo, a definição adotada no projeto ISAM é descrita na Seção 3.5 e 3.6. Numa análise preliminar, o contexto pode ser determinado pela informação de quem (entidade), onde (localização), quando (tempo), o que está fazendo (atividade) e com o que está fazendo (recurso).

Os elementos computacionais representam o conjunto de informações sobre a disponibilidade e acessibilidade das entidades nas quais a aplicação está interessada, e que influenciam seu comportamento ou desempenho. A reação às alterações no estado dessas entidades pode ser implementada usando várias técnicas e mecanismos. Em geral, alterações paramétricas, alterações de conteúdo ou formato, e alterações na funcionalidade são usadas para tal fim.

As questões relativas a "quando" e "onde" ocorre a adaptação são relativas aos aspectos de implementação da adaptação, e não são considerados na taxonomia, uma vez que influenciam no desempenho da aplicação e não na sua funcionalidade. "Quando" relaciona-se ao conceito de agilidade do sistema (NOBLE, 2000). "Onde" depende das características dos equipamentos do sistema. Dependendo da capacidade do equipamento, os mecanismos de adaptação podem se mais flexíveis e atuar em ambos, nodos móveis e estacionários.

2.3.3 Tipos de Aplicações Móveis

A combinação dos aspectos da taxonomia mostra as características de muitos domínios de aplicações móveis. Para identificá-los, usa-se o termo *consciente de x* (*x-aware)*, onde *x* identifica o principal elemento computacional que afeta o comportamento da aplicação. Assim, identificaram-se as principais categorias de aplicações *x-aware* e suas características. O nome da categoria identifica a sensibilidade ao maior *container* de elementos computacionais (nó, rede, recurso, localização, contexto, situação) que determina o ambiente de uso e execução da aplicação (contexto). As principais categorias são relacionadas abaixo. Note que cada categoria inclui as anteriores em uma seqüência de complexidade, a qual reflete o aumento da sensibilidade a elementos internos/externos que são relevantes para a aplicação.

2.3.3.1 Aplicações Nômades

Executam em um equipamento móvel, como PDA's e celulares. O domínio dessas aplicações são a administração pessoal e o acesso a informações em geral (www.palmos.com). Estas aplicações não apresentam um comportamento adaptativo, embora sejam projetadas com economia no uso dos recursos dos dispositivos portáteis (armazenamento, memória e energia); logo, não pertencem à taxonomia para aplicações adaptativas apresentada.

2.3.3.2 Aplicações Conscientes da Rede (network-aware)

Estas aplicações adaptam-se às trocas na disponibilidade dos recursos da rede, em especial à largura de banda. Elas se adaptam para fornecer uma aparência de serviço contínuo, e para gerenciar o balanceamento entre transparência, flexibilidade e uso eficiente da rede. Em muitas destas aplicações (ANGIN et al., 1998; BAGGIO, 1998; BOLLIGER; GROSS, 1998; CYBENKO; CARIPE; MOIZUMI; GRAY, 1998; DEWITT et al., 1998; FOX et al., 1998; LOWEKAMP et al., 1999; STEENKISTE,

1999) a adaptação é relativa à redução dos dados, à transformação ou às técnicas de filtragem para o transporte dos dados na rede. Nestes sistemas, a localização dos componentes é fixa, a adaptação advém de trocas no uso da rede. Em geral, essas aplicações usam mecanismos automáticos de adaptação, tais como *caching* e replicação, que podem usar técnicas de adaptação baseadas somente em parâmetros ou em troca do conteúdo. Fornecem adaptação baseada em recursos, onde o recurso considerado é a rede. O foco destas aplicações é a portabilidade e a conectividade.

2.3.3.3 Aplicações Conscientes dos Recursos (resource-aware)

Estas aplicações tentam identificar e usar os recursos locais disponíveis (NOBLE, 2000; RANGANATHAN; ACHARYA; SALTZ, 1997). A maior parte do processo de adaptação pode ser executado automaticamente pelo sistema usando ajuste de parâmetros ou alteração no formato dos componentes. Por exemplo, a indisponibilidade de uma base de dados requisitada pode forçar a aplicação a migrar para outro sítio onde este recurso está disponível. Aplicações baseadas na tecnologia de agentes móveis (GRAY et al., 1997; PEINE; STOLPMANN, 1997; PHAM; KARMOUCH, 1998) têm este comportamento. Assim, migração e descoberta de recursos são características importantes para estas aplicações. O foco destas aplicações é a mobilidade e a conectividade.

2.3.3.4 Aplicações Conscientes da Localização (location-aware)

Estas aplicações têm seu comportamento determinado pela localização física do usuário (COMPUTER, 2001). Usam o conhecimento da localização como fundo para a resposta a consultas mais especializadas, ou para executar ações dependentes da localização, ou ainda para fornecer informações sobre a área ou objetos ao redor. As características importantes para esta categoria são determinar a posição do usuário e associar dados à localização (*content-based adaptation*). Como a localização é a chave para determinar o contexto ao redor da aplicação, esta categoria é um subconjunto das aplicações conscientes do contexto.

*2.3.3.5 Aplicações Conscientes do Contexto (*context-*aware)*

Estas aplicações podem deduzir o estado interno ou externo, usando sensores ou monitores, e se ajustar ao novo estado. Também introduzem a dimensão pessoal às aplicações móveis. Estas aplicações, em geral, são personalizadas pelo usuário, e seu comportamento é baseado em onde elas estão sendo executadas (localização). Podem também tirar vantagem das computações próximas e dos recursos da vizinhança. O contexto descreve informações sobre localização, dispositivos, perfis de equipamentos e da rede, atividades, objetos computacionais (disponibilidade, confiabilidade, segurança) e outros (CHEN; KOTZ, 2000; DEY; ABOWD, 2000a). Assim, monitoramento do contexto, modelagem da informação do contexto e notificação das alterações são características importantes para estas aplicações estabelecerem um comportamento adaptativo. Para estas, também é necessário fornecer (re)configurações alternativas de acordo com o novo contexto. O foco destas aplicações está centrado na combinação de mobilidade, portabilidade e conectividade. Aqui, deve existir uma efetiva colaboração entre o sistema e a aplicação para a adaptação, possivelmente usando estratégias de ajuste de conteúdo ou funcionalidade.

2.3.3.6 Aplicações Conscientes da Situação (situation-aware)

Até este ponto, a perspectiva da adaptação é focada no desejo da aplicação de ajustar-se ao ambiente corrente – seu contexto de execução. Além disso, a decisão da adaptação é interna à aplicação, e executada pela própria aplicação ou pelo sistema de suporte (ou ambos). Outra perspectiva é introduzida quando o ambiente altera a funcionalidade da aplicação em execução, baseado na consciência das outras aplicações em execução no mesmo ambiente (situação). A decisão de adaptação é externa à aplicação, e pode ser realizada pelo sistema em colaboração com as aplicações vizinhas. Este comportamento categoriza as aplicações conscientes da situação. A situação adiciona uma nova dimensão ao contexto: a social, que reflete as capacidades dos usuários, suas aplicações e preferências. A adaptação ocorre na funcionalidade dependendo do uso do contexto e da preferência dos usuários. A característica base é a pró-atividade do sistema de gerenciamento. Um exemplo desse tipo de aplicação é dado a seguir. Um usuário está em casa, lendo um jornal em seu dispositivo móvel. Quando este se desloca para o carro, ele deseja alterar automaticamente para a tarefa que informa as condições de tráfego ou condições do tempo. Esta categoria apresenta o maior nível de abstração dentre as aplicações móveis, e ainda não existem trabalhos que abordam este domínio de aplicações.

Como o objetivo do projeto ISAM é fornecer uma infra-estrutura de software a qual permite explorar a implementação e execução de aplicações *pervasivas*, onde as pessoas possam usar diferentes dispositivos móveis em qualquer lugar, usá-los em vários ambientes para acesso à informação e executar diferentes tarefas sem a restrição de tempo e localização – identificou-se, a partir desta taxonomia, alguns requisitos para a arquitetura de software, relativos às questões: porquê, quem e com o quê, quando, onde e como ter um comportamento adaptativo ao contexto (AUGUSTIN, 2001). Isto permitiu a definição do modelo de adaptação a ser adotado na arquitetura (Capítulo 3).

2.4 Requisitos dos Sistemas Móveis Adaptativos

Em termos de Computação *Pervasiva*, o desafio é o suporte aos usuários que se deslocam, durante o transcurso da aplicação, mantendo a interação destes com o sistema em um nível satisfatório. Supondo a existência de uma infra-estrutura de base que forneça portabilidade, mobilidade e conectividade em uma escala global, acredita-se que é possível construir sistemas móveis, satisfatoriamente, se estes aderem a cinco princípios: nome global; acesso global; comunicação assimétrica, anônima e assíncrona; computação distribuída e adaptabilidade ao contexto.

Nesta perspectiva, para auxiliar a responder a questão da adaptação ao contexto, identificaram-se cinco requisitos que o sistema como um todo deve suportar. Tendo a adaptação como premissa, os requisitos na base da concepção da arquitetura ISAM são: contextualização e personalização; adaptação multidimensão e multinível; adaptação negociada; desacoplamento temporal e espacial; e funcionalidade selecionada pelo contexto.

2.4.1 Contextualização e Personalização

Entendendo o que é contexto e de que forma este pode ser usado, os projetistas de aplicações podem mais facilmente determinar que comportamentos ou características desejam que as aplicações suportem, e que elementos do contexto são requeridos para alcançar tal comportamento.

A captura do estado do contexto deve ser independente da aplicação em si, pois este se refere ao ambiente em que a aplicação executa. Adicionada às vantagens de gerenciamento global de recursos, esta independência permite à aplicação um comportamento adaptativo em tempo de carga (*startup*), seguido por um comportamento adaptativo continuado e dinâmico em relação ao contexto de execução corrente.

A individualização, por sua vez, identifica quais elementos do contexto geral são de interesse de cada aplicação, e mantém informações específicas para cada uma. Estas informações contêm o estado do contexto na perspectiva da aplicação (informações filtradas, interpretadas, refinadas). Isto permite uma flexibilidade de projeto, onde cada aplicação especifica uma visão particular de contexto ao qual se adaptará.

O sistema deve exportar informações sobre o estado dos elementos do contexto para que a aplicação possa implementar sua própria estratégia de tratar com alterações nas condições ambientais (consciência do contexto). Ao mesmo tempo, o sistema deve fornecer primitivas que simplifiquem a tarefa de reação a essas mudanças.

Este requisito está concretizado no serviço de reconhecimento de contexto da arquitetura – `ISAMcontextService` (ver Apêndice B).

2.4.2 Adaptação Multinível e Multidimensão

Para ser flexível, o sistema deve suportar uma larga variedade de elementos que constituem o contexto, os quais podem ser de diferentes naturezas: localização, tempo, largura de banda, preferência do usuário, padrão de deslocamento do usuário, etc. Estas multidimensões (físicas, lógicas e pessoais) devem ser tratadas em multiníveis (rede, sistema e aplicação) para quebrar sua complexidade.

Idealmente, a aplicação móvel deve oferecer todos os níveis de adaptação: (*i*) básico (*resource-awareness*), relativo a mudanças em recursos físicos, como banda e latência da rede; (*ii*) intermediário (*context-awareness*), o qual permite alterações no comportamento mas mantém a funcionalidade da aplicação, desde que esta pode alterar não somente a qualidade da informação, mas também seu conteúdo (por exemplo, objetos sensíveis à localização são somente relevantes em lugares particulares); (*iii*) alto nível (*situation-awareness*), relativo às mudanças das situações/comportamento do usuário que se reflete na alteração da funcionalidade da aplicação. Nos sistemas atuais, somente o nível básico está normalmente presente nas arquiteturas de software investigadas. O ISAM atende aos níveis básico e intermediário.

Este requisito está concretizado no modelo de adaptação da arquitetura ISAM (ver Capítulo 3).

2.4.3 Adaptação Negociada

Considerando que aplicações móveis devem ser adaptativas por natureza, estas podem requerer atuar e ter consciência sobre vários aspectos da computação, tais como: replicação, contexto de execução, serviços e localização. Para alcançar esta consciência, a aplicação ou o sistema deve pró-ativamente monitorar e controlar as condições do ambiente, e a aplicação e/ou o sistema devem reagir às alterações através do processo de adaptação.

Considerando a adaptação no nível de aplicação, esta é responsável por tratar com freqüentes desconexões, perda de pacotes, alteração na banda, etc. Nesse caso, não existe suporte do sistema de execução no ambiente móvel. Aumento da eficiência resultante de protocolos de adaptação mais específicos é a vantagem dessa solução. O sistema pode ser deixado intacto, sem nenhuma alteração, já que a responsabilidade é deslocada para o lado da aplicação. Existem dificuldades, entretanto, na implementação

da adaptação no lado da aplicação resultante de muitos problemas a serem tratados. Outra conseqüência é o aumento do custo de desenvolvimento considerando que muitas aplicações, existentes ou novas, podem requerer um comportamento adaptativo.

Na adaptação no nível do sistema, o sistema de execução assume a responsabilidade de tratar com o ambiente móvel, sendo esta transparente para a aplicação. A maior vantagem é que as aplicações pré-existentes não necessitam serem modificadas, a princípio. Porém, essas aplicações, originalmente desenvolvidas para a rede fixa, podem não ser adequadas para a mobilidade. O principal problema é tentar implementar adaptações eficientes e úteis, uma vez que muitas das limitações do ambiente móvel podem ser específicas de uma dada aplicação. Desenvolver adaptação no nível da aplicação é uma tarefa complexa, pois o desenvolvedor também deve desenvolver o software para observar o ambiente. Por outro lado, a adaptação realizada pelo sistema de forma transparente é de orientação geral, o que pode torná-la inadequada para uma aplicação específica. Conclui-se que, pela natureza pessoal, heterogênea e dinâmica da mobilidade, o sistema de suporte não deve tomar todas as decisões independentemente da aplicação.

Segundo Noble (2000), a adaptação às variações no contexto de execução é mais bem atendida se a decisão de adaptação for realizada através de uma colaboração entre a aplicação e o sistema. Este último gerencia o sistema de execução como um todo e é, portanto, a parte mais indicada para saber o contexto e tomar decisões globais mais eficientes. Por outro lado, a aplicação conhece melhor seu domínio e semântica e pode atuar mais eficientemente nas decisões de âmbito específico/local. Nota-se, então, que é necessário um equilíbrio negociado entre as necessidades e interesses de uma aplicação específica e as necessidades do sistema em geral. O desafio é construir sistemas que tenham o equilíbrio entre essas duas características.

Este requisito embasa o modelo de adaptação colaborativa entre a aplicação e o *middleware*, definido no ISAM e detalhado no Capítulo 3.

2.4.4 Desacoplamento Temporal e Espacial

Muitas atividades são distribuídas no tempo e no espaço e entre os usuários móveis. Porém, as restrições naturais da Computação Móvel não garantem a interação contínua dos integrantes da aplicação distribuída. Desconexões são comuns no ambiente, tanto devido ao meio físico quanto à economia de energia. Assim, o ambiente móvel requer mecanismos de coordenação, para comunicação e sincronização, entre os componentes sem a premissa da conexão permanente. Além disso, pela natureza imprecisa do ambiente móvel, o modelo de coordenação deve enfatizar a comunicação assíncrona.

Uma forma de obter este desacoplamento espacial e temporal é utilizando-se uma abstração semelhante ao Espaço de Tuplas do modelo Linda (GELERNTER, 1985) na forma de múltiplos espaços. Esse modelo torna possível que produtor e consumidor de tuplas não coexistam no tempo e no espaço – importante para o ambiente onde não há garantias de conectividade. Esta solução também permite separar dados de funcionalidade (código), de forma que estes podem ser gerenciados separadamente e evoluir independentemente. Isto é importante para serviços que pesquisam, filtram ou traduzem dados. Ao mesmo tempo, dados e código dependem um do outro, para a migração da aplicação, por exemplo. Desta forma, o sistema deve incluir a habilidade de agrupar dados e de acessá-los independentemente. Espaço de Tuplas usa por definição a estratégia *pull* de entrega de dados (sob demanda, operação *out*). É necessário expandir o modelo para permitir a semântica *push* de entrega de dados (disseminação), para componentes que registram interesse numa funcionalidade dirigida por eventos (*event-driven)*, na forma de tupla reativa (AUGUSTIN, 2000).

Este requisito define o modelo de comunicação e sincronização adotado na arquitetura ISAM: espaço de objetos com natureza reativa (YAMIN, 2004).

2.4.5 Funcionalidade Selecionada pelo Contexto

A aplicação móvel deve ser desenvolvida com adaptação e mobilidade em mente para ser capaz de comportar-se de forma otimizada no ambiente móvel e beneficiar-se do contexto (COUDERC; KERMARREC, 1999). Além disso, a estrutura da aplicação deve facilitar seu comportamento adaptativo, selecionado automaticamente pelo sistema com base na demanda de recursos disponível e requerida pela aplicação. A adaptação é assíncrona e imprevisível, desta forma requer que o programador modele o comportamento da aplicação com diferentes níveis de demanda de recursos. O sistema, por sua vez, deve monitorar os recursos e fazer um chaveamento automático para o comportamento adequado ao consumo de recursos correntemente disponíveis. Ou seja, a gerência da adaptação deve ficar a cargo do sistema de suporte. Além disso, o sistema deve facilitar a composição e a expansão das aplicações e serviços em tempo de execução. Deve ser possível alterar o comportamento da aplicação ou adicionar novo comportamento sem alterar a aplicação em si.

Este requisito é a base para a definição do modelo de adaptação no nível da aplicação (ver Capítulos 3 e 4).

Por fim, salienta-se que esses cinco (5) requisitos estão na base tanto da concepção e codificação da aplicação ISAMadapt quanto nos componentes e serviços do *middleware* EXEHDA. Os próximos Capítulos abordam como a arquitetura ISAM foi organizada para atender a estes requisitos.

3 ARQUITETURA ISAM

Acredita-se que, em vez de o sistema mascarar a mobilidade (transparência), como é característica dos sistemas distribuídos, esta deve ser explorada. Para tal, deve ser fornecida uma infra-estrutura para suportar novos tipos de aplicações conscientes da mobilidade, que permitam ao programador tirar vantagem das características dinâmicas do ambiente, porém sem consumir muito de sua atenção.

Neste sentido, o projeto ISAM[6] (AUGUSTIN, 2002; AUGUSTIN, 2001) (Infra-estrutura de Suporte às Aplicações Móveis Distribuídas) iniciou um estudo das principais questões envolvidas na modelagem de aplicações móveis conscientes do contexto em um ambiente de Computação *Pervasiva* com objetivo de construir uma infra-estrutura com suporte para a adaptabilidade. Foram identificadas as características e requisitos das aplicações para atender a demanda deste novo ambiente (AUGUSTIN, 2001), resumidas nas Seções 2.3 e 2.4. Como a infra-estrutura necessária para suportar a Computação Móvel em escala global não está ainda disponível, o produto final deste estudo foi o projeto de uma arquitetura de software que explora alternativas para simplificar a modelagem, programação e execução de aplicações móveis com comportamento adaptativo, conscientes do contexto, em um ambiente *pervasivo*. Este Capítulo aborda os tópicos e as questões envolvidas na modelagem da arquitetura ISAM.

3.1 Novo Modelo de Aplicação

Hoje, as aplicações projetadas para equipamentos portáteis focalizam o domínio de administração pessoal, e começam a permitir o acesso sincronizado a dados residentes em servidores. Estas aplicações estão sendo direcionadas para um uso empresarial, mas ainda mantendo um comportamento nômade. Um usuário de dispositivo móvel vê este dispositivo como um mini *desktop*, e as aplicações como programas que executam nele e acessam código e dados localmente. Dados remotos são atualizados via um processo que utiliza um software de sincronização, por exemplo o HotSync da Palm Computing.

Porém, a **mobilidade do usuário** requer um novo modelo de aplicações que vê o **poder computacional espalhado em toda a rede**, e não residente em um dispositivo que tem a capacidade esporádica de comunicação via um link sem fio, e que armazena e executa software. Na visão ISAM, o computador é a rede em uma grade global. Os componentes do ambiente computacional (dados, código, vários tipos de dispositivos, serviços e recursos) estão espalhados, e são gerenciados por um sistema de suporte que fornece um acesso *pervasivo* a eles. Cada usuário tem um **ambiente virtual** que pode

[6] Projeto parcialmente financiado pela FAPERGS – Fundação de Amparo a Pesquisa do Rio Grande do Sul, aprovado no edital 003/2001. O projeto é um consórcio de quatro universidades do sul do país: UFSM, UFPel, UFRGS e UNISINOS. Informações na página www.inf.ufrgs.br/~isam.

ser acessado na localização em que se encontra e com o equipamento de que este dispõe. As aplicações são distribuídas, móveis e adaptativas ao contexto, tendo uma **semântica siga-me**. O código das aplicações é enviado **sob demanda** aos dispositivos, e **adaptado ao contexto** corrente onde eles se encontram. Os dispositivos móveis são como **dispositivos de interface**, não armazenam código nem dados permanentemente, funcionam como **portais** que recebem código para executar e podem transferir a execução para máquinas mais próximas ou com melhores recursos. O código da aplicação deve explorar as capacidades dos dispositivos e de toda a rede que compõe o ambiente computacional do usuário. Alterações no ambiente, devido à dinamicidade deste e ao deslocamento do usuário, devem ser refletidas no comportamento das aplicações. As **aplicações são gerenciadas** pelo sistema de suporte, que é orientado por políticas definidas na etapa de desenvolvimento. A Figura 3.1 ilustra esse ambiente.

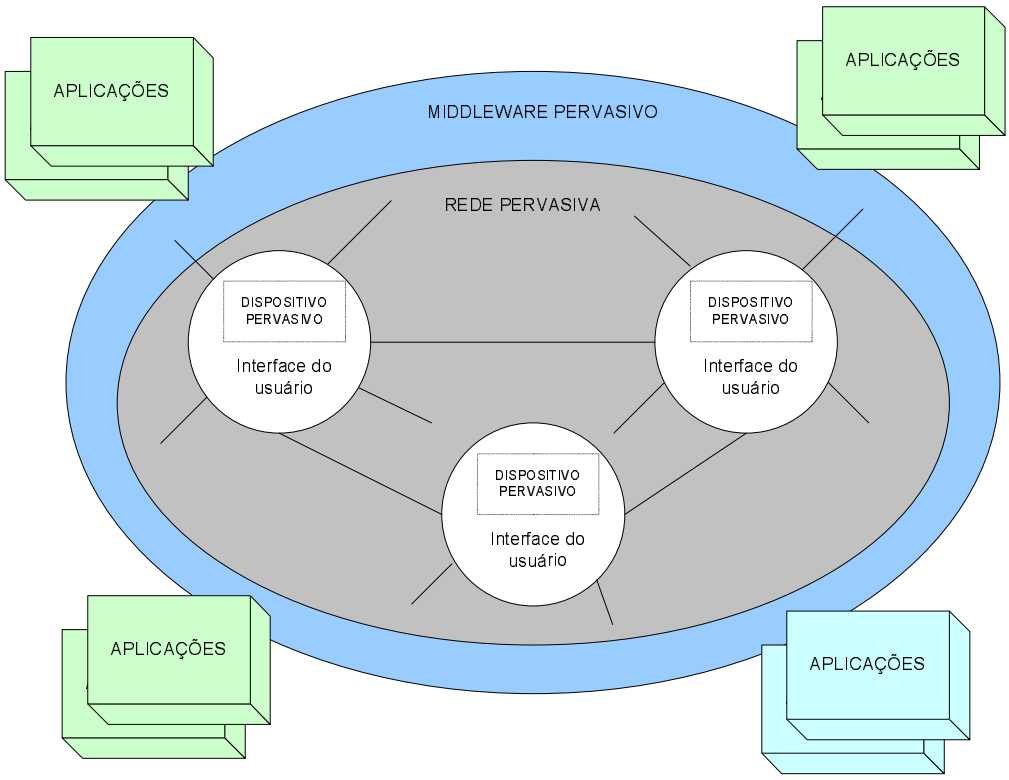


Figura 3.1: Visão do Ambiente *Pervasivo*

Com o objetivo de disponibilizar estas funcionalidades, foi projetada uma divisão de responsabilidades entre o projetista de aplicações e o sistema de execução, implementado na forma de um *middleware*. Desta forma, ISAMadapt – ambiente de desenvolvimento, e EXEHDA – ambiente de execução, têm um alto grau de integração e acoplamento.

3.2 Arquitetura ISAM

Como visto, a propriedade essencial do sistema de software para Computação *Pervasiva* é sua habilidade de se ajustar às constantes alterações das condições ambientais. A aplicação ISAMadapt é uma aplicação móvel, distribuída e adaptável. Neste sentido, a **adaptação ao contexto** é um conceito-chave na arquitetura de software ISAM.

Tornar uma aplicação adaptativa pode requerer uma profunda alteração na sua estrutura. Para amenizar essa exigência, algumas funcionalidades (e complexidade) associadas com a adaptação podem ser embutidas em sistemas de suporte. Desta forma, a arquitetura ISAM, modelada com este objetivo, apresenta uma organização lógica em três camadas: (i) camada de aplicação; (ii) camada de suporte e ambiente de execução; (iii) camada de sistemas básicos. Na Figura 3.2, a representação da consciência do contexto como um módulo da arquitetura é virtual, está em destaque dada a sua importância, uma vez que está presente na concepção de todos os seus módulos.

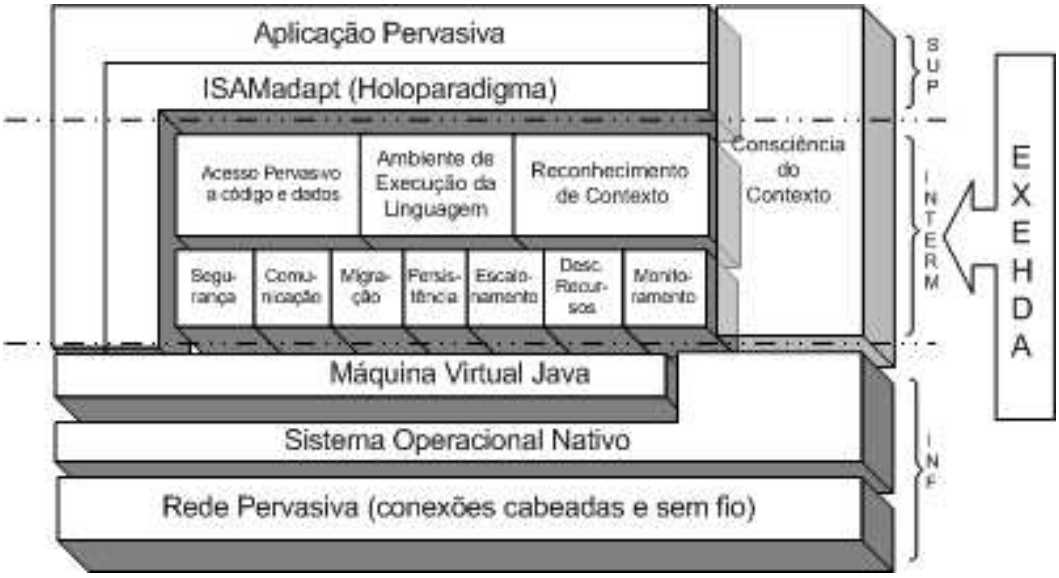


Figura 3.2: Visão Geral da Arquitetura ISAM

A camada superior (aplicação) oferece o paradigma (HoloParadigma) e a linguagem de programação (ISAMadapt), bem como a forma explícita de programar aplicações *pervasivas* (móveis, distribuídas e conscientes do contexto).

A camada *middleware* (EXEHDA) oferece os mecanismos de suporte à execução da aplicação *pervasiva* e às estratégias de adaptação (YAMIN, 2004). Além disso, utiliza primitivas de baixo nível (subprojeto PRIMOS) desenvolvidas para acomodar suas necessidades (SILVA, 2003). Esta camada tem dois níveis lógicos.

O primeiro nível é composto por três módulos de serviço à aplicação: Suporte à Pervasividade, Reconhecimento de Contexto e o 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), Base de Dados Pervasiva (ISAMbda). O Ambiente Virtual do Usuário compõe-se dos elementos que integram a interface de interação do usuário com o sistema. Este módulo é o 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. Este módulo deve caracterizar, selecionar e apresentar as informações, de acordo com as necessidades e o contexto em que o usuário se encontra. Para realizar estas tarefas, o sistema se baseia num modelo de uso, onde as informações sobre o ambiente de trabalho, preferências, padrões de uso, padrões de movimento físico e hardware do usuário são dinamicamente monitoradas e

integram o Perfil do Usuário e da Aplicação. Esse componente da arquitetura ainda não foi modelado, somente uma simplificação necessária para os demais componentes está sendo implementada (YAMIN, 2004).

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. Este serviço, chamado `ISAMcontextService`, é detalhado no Apêndice B.

O **Ambiente de Execução da Linguagem**, por sua vez, é o componente-chave do gerenciamento da adaptação na arquitetura ISAM. Aspectos da sua funcionalidade são descritos nos Capítulos 4 e 5.

No segundo nível da camada intermediária estão os serviços básicos do ambiente de execução ISAM que provêem a funcionalidade necessária para o primeiro nível e cobrem vários aspectos, tais como migração – mecanismos para deslocar um componente de uma localização física para outra (SILVA, 2003); persistência – mecanismo para aumentar a disponibilidade e o desempenho do acesso aos dados; localização e *naming* – para dar suporte ao movimento dos dispositivos móveis e dos componentes entre diferentes células, mantendo a execução durante o deslocamento, e para a descoberta de recursos; comunicação – implementada através de multiespaços de objetos distribuídos com suporte a tupla de natureza reativa (YAMIN, 2004); escalonamento – permite decidir o melhor nodo para criar os componentes da aplicação (REAL, 2003); monitoramento – sensores que fornecem informações sobre o ambiente de execução e a aplicação (SILVA, 2003).

A camada inferior da arquitetura é composta pelos sistemas e linguagens nativas das máquinas que integram a rede móvel. Por questões de portabilidade, nesta camada a plataforma base de implementação é a Máquina Virtual Java em suas diferentes versões: J2SE e J2ME (ver Apêndice A). A arquitetura supõe a existência de uma rede *pervasiva*, que é móvel, infra-estruturada e que dê suporte de acesso global aos usuários.

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, e atendimento aos requisitos expostos na Seção 2.4. 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. A arquitetura proposta é abrangente, o espaço de atuação do projeto ISAM é amplo. Neste momento, o foco dos trabalhos concentra-se na implementação dos principais mecanismos da arquitetura ISAM, explorando a adaptação ao contexto em termos das abstrações para expressá-la no nível de linguagem de programação (ISAMadapt) e dos mecanismos necessários para sua execução (*middleware* EXEHDA).

3.3 Modelo de Rede

O modelo de rede móvel adotado para o ambiente *pervasivo* ISAM (ISAMpe) é caracterizado como rede infra-estruturada[7], a qual compõem-se de várias **células de execução** (`EXEHDAcell`). Cada célula de execução pode ser composta por: `EXEHDAmob-node` (dispositivos portáteis móveis que se comunicam por meio sem fio) com acesso de comunicação às `EXEHDAbases` (servidores de rede com interface para redes sem fio), e por `EXEHDAnodes`, com acesso cabeado (*wired*) à base (Figura 3.3). A estrutura está organizada de forma que o dispositivo móvel se comunica com uma `EXEHDAbase`, que oferece um contínuo ambiente de processamento e

[7] Alguns autores nomeiam esta rede como Rede Móvel Nômade.

comunicação para os nodos móveis registrados. As EXEHDAbases estão ligadas entre si por uma rede fixa, permitindo o acesso indireto dos EXEHDAmob-nodes a toda estrutura de rede. O EXEHDAmob-node pode se deslocar fisicamente dentro de uma EXEHDAcell (área de abrangência da comunicação com a EXEHDAbase), a qual pode variar de dimensões: pico, micro, macro, e entre as EXEHDAcells. Fisicamente a célula se refere à área de abrangência de um Ponto de Acesso na Rede Sem Fio Local (padrão IEEE 802.11) ou à área de acesso da comunicação celular.

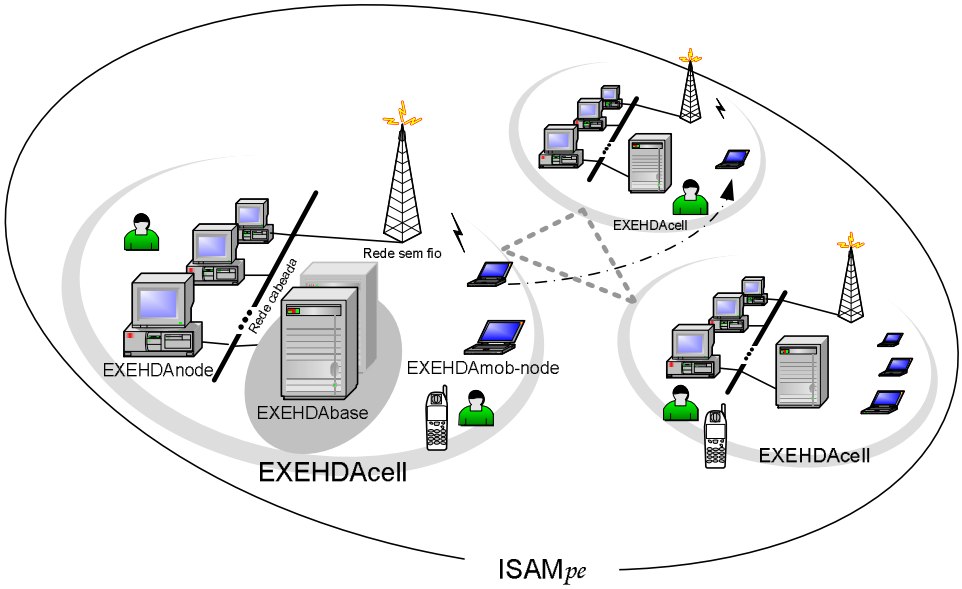


Figura 3.3: Componentes do Gerenciamento Físico da Arquitetura ISAM

3.4 Adaptação na Arquitetura ISAM

A adaptação não é uma propriedade funcional da aplicação. Isto induz à decisão de que sua concepção pode ser realizada em uma etapa separada. Durante o projeto de um sistema adaptável, é necessário identificar e qualificar as características que representam aspectos importantes para que decisões de adaptabilidade possam ser tomadas. É importante notar que no modelo ISAM, nesse momento, 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 é crítico. A seguir, detalham-se os principais aspectos envolvendo o conceito de adaptação modelados no ISAM.

3.4.1 O Processo de Adaptação

A modelagem do processo de adaptação no ISAM, esquematizada na Figura 3.4, foi dividida em etapas:

- 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;
- 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;
- ativação da ação, refere-se à execução da ação adaptativa selecionada.

Cada etapa é implementada por um componente da arquitetura ISAM, a saber: `ISAMcontextService`, `ISAMadaptEngine` e `EXEHDA`, conforme detalhado no Capítulo 4.

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) fornecer a funcionalidade conectada em face à desconexão utilizando *buffering* e *prefetching.*

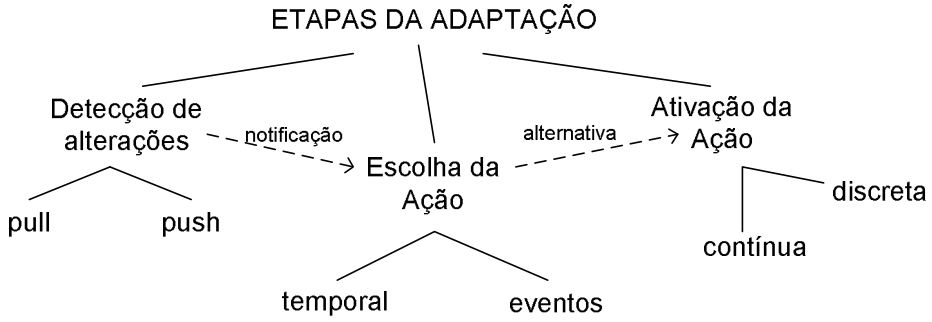


Figura 3.4: Etapas da Adaptação ISAM

O conjunto de conceitos usados para tratar adaptação no ISAM é mostrado na Figura 3.5. Os conceitos são derivados do suporte necessário para tratar a adaptação sob três aspectos: as informações para adaptação derivadas do ambiente; o gerenciamento da execução da adaptação; e a expressão do comportamento adaptativo.

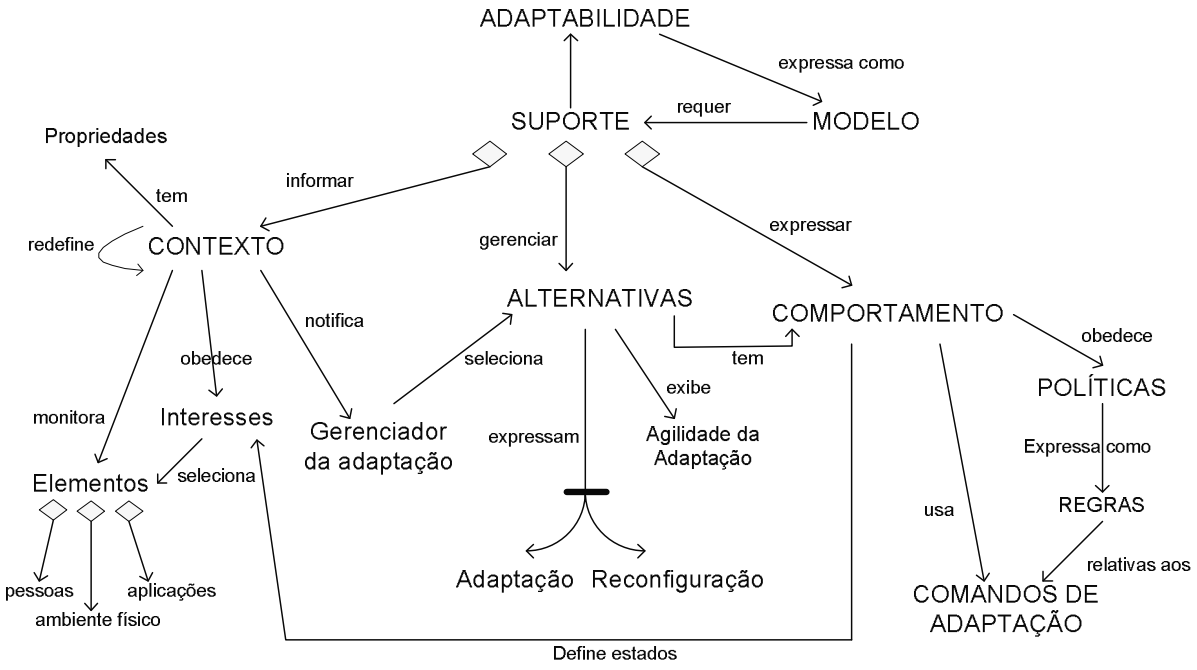


Figura 3.5: Modelando a Adaptação ISAM

Considera-se que as abstrações de sistemas operacionais existentes não são suficientes nem apropriadas para tratar as necessidades das aplicações *pervasivas.* Defende-se um estilo de programação que força a aplicação a exibir explicitamente um

comportamento adaptativo que será gerenciado e executado por um ambiente de suporte (*middleware*).

A análise dos vários sistemas móveis adaptativos e aplicações-protótipo, com suas variadas noções de adaptação (ver Capítulo 7), permitiu identificar os **requisitos de propósito-geral para expressar adaptabilidade ao contexto no nível de linguagem** de forma genérica. Estes requisitos incluem:

a) descrição do elemento de contexto de interesse – identificação dos elementos que compõem o ambiente e que modelam uma visão particular de contexto;
b) descrição do comportamento adaptativo – coleções de ações e 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);
c) descrição de políticas – regras de propósito geral ou particular que orientam as decisões de adaptação realizadas pelo *middleware*.

Estes requisitos tornam-se operacionais através de um conjunto mínimo de primitivas para atender a expressividade, relativos a contexto, adaptadores e mecanismos de adaptação, implementadas na linguagem ISAMadapt. A programação é facilitada pelas interfaces de programação do ambiente de desenvolvimento e o suporte dado pelo *middleware*. Detalhes são discutidos no Capítulo 4 e na referência (YAMIN, 2004), respectivamente.

3.4.2 O Modelo Colaborativo

Diferente de sistemas onde a adaptação toma lugar em tempo de compilação (KELEHER; HOLLINSWORTH; PERKONIC, 1999) ou de carga (KRAUTER; BUYYA, 2001), a adaptação ISAM deve ser dinâmica, ocorrendo enquanto o sistema está sendo realmente usado; automática, quando possível; e com colaboração da aplicação, quando necessário. Para abordar a complexidade da adaptação dinâmica, ISAM usa um modelo colaborativo no qual sistema e aplicação contribuem entre si para a realização da adaptação. A colaboração é gerenciada pelo *middleware* de execução. Para expressar a colaboração, a aplicação ISAMadapt decide como se adaptar a uma situação específica, e a expressa na forma de **políticas de adaptação**, que orientam o EXEHDA na tomada de decisão para execução da adaptação. A flexibilidade é alcançada pelas abstrações da linguagem e respectivo suporte *run-time*.

Diferente dos outros sistemas móveis (ver Capítulo 7), a adaptação não é totalmente transparente para a aplicação. Aplicações podem controlar o processo dependendo do seu propósito. Para isto, informações de contexto são necessárias. Tipicamente, as aplicações necessitam conhecer a localização do usuário e dos equipamentos, as capacidades e disponibilidades destes equipamentos e da infra-estrutura de rede.

Assim, a arquitetura de software ISAM adota a estratégia denominada **Adaptação Colaborativa Multinível** (Figura 3.6), onde:

a) o *middleware* (EXEHDA) é responsável por adaptações, normalmente, relacionadas ao desempenho e à gerência de recursos do ambiente, como a re-locação de componentes;
b) a aplicação ISAMadapt é responsável por decisões de adaptações específicas de seu domínio e situações de uso;
c) ambos são responsáveis por uma negociação de decisões de adaptação.

Muitas abordagens enfatizam a transparência do gerenciamento dos aspectos não-funcionais, como adaptação. Porém, considera-se que esta abordagem caixa-preta não é apropriada para o desenvolvimento de aplicações *pervasivas* com o grau de flexibilidade e generalidade que propõe o projeto ISAM. Desta forma, a proposta para

minimizar o conflito entre transparência e consciência dos aspectos da adaptação é a colaboração entre aplicação e sistema. Adaptações desta natureza permitem uma implementação mais eficiente resultando em aplicações mais robustas e aumentando o grau de aceitação do usuário (NOBLE, 2000). O sistema fornece o suporte para as aplicações, fornecendo serviços especiais não disponíveis no modelo de adaptação no nível da aplicação. O problema a resolver é o nível de cooperação requerido entre projetistas de sistema e projetistas de aplicações para criar protocolos aceitáveis para ambos os grupos. Também, o custo crescente de desenvolvimento pode ser inicialmente alto desde que adaptações podem ser requeridas em ambos os lados.

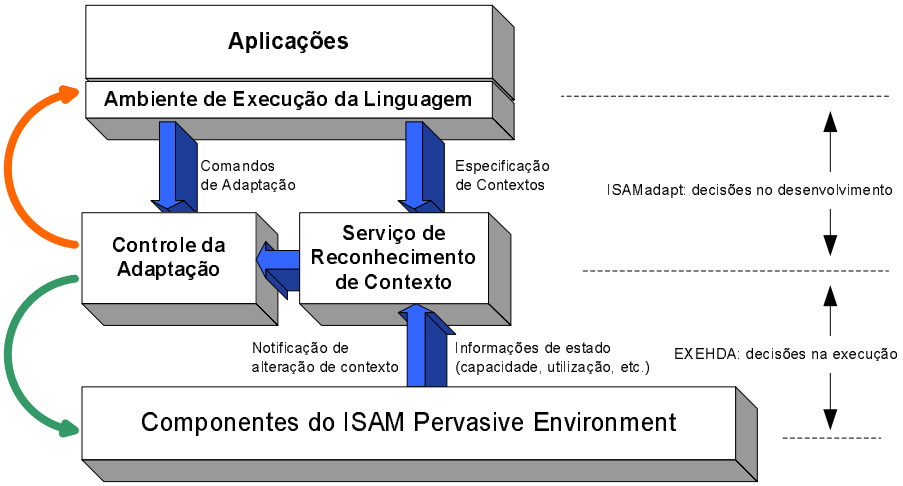


Figura 3.6: Modelo Colaborativo de Adaptação ISAM

O comportamento colaborativo da adaptação na arquitetura ISAM é disponibilizado através do ambiente de execução que gerencia os aspectos da adaptação, EXEHDA, e um ambiente de programação que permite expressar as necessidades adaptativas da aplicação, `ISAMadapt IDE`. O comportamento colaborativo-adaptativo do ISAM permite múltiplas possibilidades. Para ilustrá-lo, apresentam-se algumas possibilidades de adaptação em dois momentos: tempo de carga e tempo de execução.

3.4.2.1 Adaptação Colaborativa em Tempo de Carga

ISAM mantém informações sobre os usuários e suas aplicações no Ambiente Virtual do Usuário (AVU), e no Ambiente Virtual de Aplicações (AVA), respectivamente. Informações, no AVU, incluem identificação do usuário (*login*, senha, *home*), aplicações instaladas, dados do usuário, última aplicação executada, último ponto de contato com a rede (`EXEHDAnode`), perfil do usuário e suas preferências. Quando o usuário conecta-se à rede, o procedimento de *login* contacta o `AVUmanagement Service` para obter o último AVU armazenado. Este é apresentado ao usuário após sofrer algumas adaptações. As adaptações realizadas são baseadas na capacidade do dispositivo móvel sendo usado, na qualidade da conexão, no perfil do usuário e suas aplicações. De acordo com a possibilidade, é enviado o AVU completo ou somente os componentes básicos. O último AVU é encontrado no último ponto de contato do usuário com a rede (`EXEHDAbase`). Se este não for localizado, será buscado a partir do nodo `EXEHDAhome` do usuário, o qual atua como um *backup* das informações do usuário no sistema. Conforme aumenta a qualidade da conexão, o `AVUmanager` pode fazer a busca antecipada (*prefetching*) de outros componentes de acordo com o perfil do usuário e de suas aplicações.

Outro exemplo de comportamento adaptativo é o relacionado à localização de dados e código. O código da aplicação ISAMadapt é particionado em componentes (entes), os quais possuem código armazenado em uma base de dados *pervasiva*, a `ISAMbda`. Ao código dos componentes da aplicação são associadas informações de carga para o escalonador (políticas). Uma parte destas informações diz se o código deve ser enviado ao nodo móvel ou ao nodo estático, e se este deve ser alocado próximo a recursos (co-alocação). Deste modo, o escalonador pode decidir mais apropriadamente onde criar os componentes da aplicação.

3.4.2.2 Adaptação Colaborativa em Tempo de Execução

Mais tarde, quando o sistema altera seu contexto de execução, o escalonador usa as informações de contexto para dinamicamente reescalonar as aplicações. O escalonador automaticamente faz algumas adaptações, tais como migração e replicação da base de dados, disparadas por decisões de balanceamento de carga, por exemplo. A carga dos componentes (código) também é dinâmica. O escalonador pesquisa pelo código mais próximo entre os `EXEHDAnodes` que armazenam a `ISAMbda`.

3.5 Ênfase no Contexto

O ambiente móvel é rico em contexto, com usuários e dispositivos movendo-se, e serviços computacionais tornando-se disponíveis ou indisponíveis, sobre o tempo. Essas informações usualmente não são disponíveis para as aplicações, mas são úteis para modelar o comportamento da aplicação que deve ser projetada para uma variedade de ambientes, onde serviços alteram sua disponibilidade, e a forma como executam, e recursos tornam-se disponíveis ou não. Não é viável projetar uma aplicação móvel para cada ambiente possível e encarregar o usuário de selecionar e ativar a aplicação adequada ao ambiente corrente. É necessário um comportamento adaptativo – consciente do contexto - da própria aplicação.

No ambiente *pervasivo*, a mobilidade física introduz a possibilidade do movimento do usuário durante a execução da aplicação. Enquanto se movimenta, os recursos podem se alterar, não só em função da área de cobertura e heterogeneidade das redes, como em função da sua disponibilidade ser variável no tempo, devido à alta possibilidade de concentração de muitos usuários em um ponto localizado no espaço. Em conseqüência, a localização corrente do usuário determina o contexto de execução da aplicação. A Figura 3.7 esquematiza esta situação.

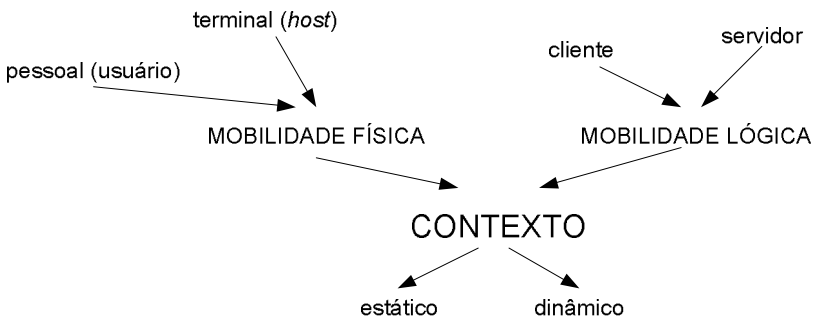


Figura 3.7: Contexto Determinado pela Mobilidade

A abstração de contexto permite focalizar em alguns aspectos que são relevantes em uma situação particular enquanto permite ignorar outros. No ISAM, contexto é definido como “toda informação relevante para a aplicação e que pode ser obtida por esta”. O

programador explicitamente identifica as entidades[8] e define seus atributos, os quais integram o contexto da aplicação (AUGUSTIN, 2001). Por exemplo, a entidade nodo poderá ter como elemento de contexto: ociosidade, carga computacional, poder computacional, número de nodos. Alterações no estado dos atributos das entidades disparam o processo de adaptação na aplicação. Assim, pode-se refinar a definição de contexto para "todo atributo de uma entidade cuja alteração em seu estado dispara um processo de adaptação na aplicação ISAMadapt".

Exemplos de elementos de contexto são representados com a metáfora de um guarda-chuva, representado na Figura 3.8, que ilustra o fato de que informações de contexto podem ser obtidas da parte fixa e móvel da rede, e de diversas e diferentes fontes: sensores de hardware, sensores de software, informações de perfil fornecidas pelo usuário. Essas informações podem ser combinadas, ou derivadas de outras informações, e constituem a origem das informações que disparam o processo de adaptação. Como informações de contexto exibem uma diversidade de características, estas devem ser estruturadas em um **modelo de contexto**.

3.6 Modelo de Contexto

Dois tipos de contexto têm sido os mais estudados: localização e comunicação (CHEN; KOTZ, 2000). Uma análise dos sistemas móveis adaptativos (Capítulo 7) permitiu esboçar um modelo de contexto que reflete características e funcionalidades necessárias para o suporte às aplicações *pervasivas*. As principais observações realizadas são registradas a seguir.

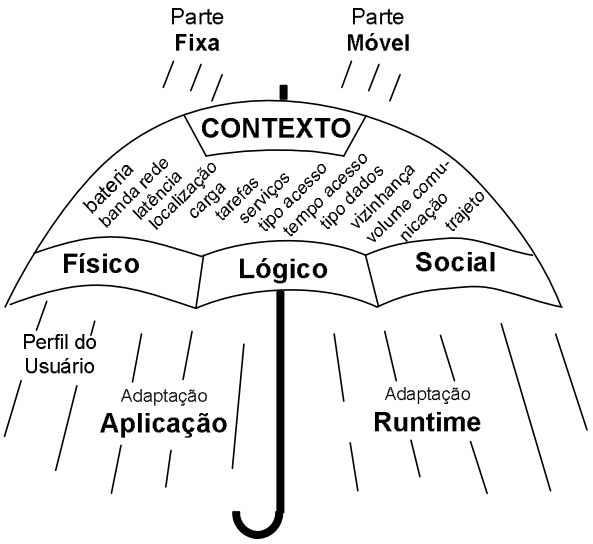


Figura 3.8: Exemplos de Elementos de Contexto ISAM

3.6.1 Observações sobre Contexto no ISAM

O contexto pode referir-se a **informações ambientais** (recursos físicos), **funcionais** (recursos lógicos) ou **comportamentais** (usuário). Estas informações exibem características temporais (variáveis no tempo) que podem ser caracterizadas como

[8] Entidades, como pessoas e nodos, não têm uma representação explícita no modelo de contexto ISAM, somente seus atributos (localização, atividades, preferências, capacidades,...) formam os elementos de contexto presentes na modelagem.

estáticas – descrevem aspectos invariantes, como tipo de dispositivo, ou **dinâmicas** – caracterizam aspectos com alterações freqüentes, como a localização do usuário. As informações estáticas podem ser obtidas diretamente com o usuário ou através de arquivos de configuração dos sistemas. As informações dinâmicas devem ser obtidas instantaneamente ou periodicamente do ambiente. Para tal é necessário um serviço de monitoramento para obter os dados de sensores tanto de hardware quanto de software. Muitos trabalhos neste campo enfocam o problema relativo aos sensores (ANGIN et al., 1998; FOX et al., 1998; LOWEKAMP et al., 1998; MILLER; STEENKISTE, 2000; ARAUJO, 2002; SILVA, 2003) e apresentam soluções de como monitorar uma variedade de aspectos das aplicações. Desta forma, o modelo de contexto ISAM pode ignorar a questão de como a informação é obtida pelo sensor, e focalizar sua atenção no tratamento e disseminação da informação de contexto dentro do espaço de aplicações.

Informações de contexto podem refletir o **estado corrente** ou serem derivadas de **informações históricas** (passado) com vistas a predizer o futuro. Para obter tal funcionalidade, o modelo deve armazenar as informações de contexto coletadas para análise/acesso futuro.

Informações de contexto podem ser imprecisas ou ficarem desatualizadas por várias razões: produtores e consumidores de informações de contexto podem estar distribuídos e distantes uns dos outros, conduzindo a um atraso entre a geração da informação e o uso desta; sensores e algoritmos para monitoração são propensos a falhas; desconexões ou falhas de conexão entre produtor e consumidor podem levar a um desconhecimento parcial ou total do contexto (*unknow* é sempre um resultado possível). Esta questão reflete a necessidade de se conhecer a **qualidade da informação obtida**.

Muitas das informações úteis à aplicação são derivadas de sensores. Porém, existe um *gap* entre a saída do sensor e o nível de informação requerido pela aplicação. Este *gap* pode ser quebrado com uma série de processamento sobre a **informação sensorada** (**filtragem, agregação, refinamento**) até transformá-la em **informação de contexto** útil à aplicação. Além disso, cada aplicação pode requerer uma **interpretação** diferente para o mesmo dado sensorado com diferentes níveis de abstração. Desta forma, a informação de contexto deve ter representações alternativas que permitam atender a este requisito.

Informações de contexto podem ser derivadas ou compostas de outras informações mais simples. Por exemplo, a atividade em execução pode ser derivada da localização corrente do usuário e de informações de atividades passadas nesta localização. Isto reflete a necessidade de estabelecer **relacionamentos** entre os elementos de contexto.

Resumindo, o modelo de contexto deve ser projetado para capturar as **informações relevantes** (elementos de contexto) para o projeto e construção de sistemas e aplicações que devem se ajustar ao contexto corrente ou previsto. Este deve prover informações de vários tipos. As informações de contexto podem ser sensoradas, derivadas ou fornecidas pelo usuário, como suas preferências. Podem ser correntes ou históricas. As informações podem ser simples – obtidas de uma única fonte, ou compostas – obtidas de várias fontes através do conceito de coleção (média, máximo, etc.), alternativas (ou, e). As informações também podem ser relativas ao tempo (temporal), a localização geográfica (espacial), ao usuário (pessoal) ou grupos de usuários (social).

O modelo ISAM considera contexto como o conjunto de dados externos que o programador decide ser o(s) elemento(s) que influencia(m) o comportamento da aplicação, e que podem ser obtidos. Logo, para uma aplicação específica somente um subconjunto do contexto disponível é capturado e usado. Logo, necessita-se de um esquema de propósito geral e extensível para descrever a informação de contexto. Esta descrição será analisada e usada para guiar o processo de adaptação. O esquema de

descrição é estabelecido por uma relação de tradução que será definida e implementada pelo projetista do sistema, orientado por um modelo (*template)*. O protocolo de entrega de informações de contexto para a aplicação é baseado em tupla com natureza reativa.

Estas observações permitiram projetar um sistema de observação e disseminação das informações de contexto que atende os requisitos das aplicações do sistema ISAM, o `ISAMcontextService` (ver Apêndice B).

3.6.2 Modelo Simplificado de Contexto

O modelo de contexto ISAM é informal, como ocorre com a maioria dos trabalhos em *context-aware applications* que desenvolvem infra-estrutura para aplicações (DEY, 2000; CHEN; KOTZ, 2000; SCHILIT; NORMAN; WANT, 1994). A informalidade do modelo justifica-se pela crença que a experimentação com aplicações modeladas informalmente permite a criação de um modelo formal mais preciso, uma vez que somente agora pesquisas começam a abordar a modelagem formal de contexto (HENRICKSEN; INDULSKA, RAKOTONIRAINY, 2002).

Pode-se começar a definir um modelo simples 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 sensorado 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 pela aplicação (visão particular).

Este modelo pode ser representado através do mapeamento para o modelo orientado a objetos usando uma notação adicional relativa ao contexto, conforme legenda da Figura 3.9, cuja representação gráfica esboça o contexto de uma aplicação. O estado do contexto é modelado como um atributo da entidade da qual é obtido. O tipo de contexto, que indica como este é obtido na visão da 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, sofisticadas o bastante para fazer as distinções necessárias para as aplicações, e simples para fornecer uma base prática para a programação. O Apêndice B detalha uma arquitetura para este *middleware*.

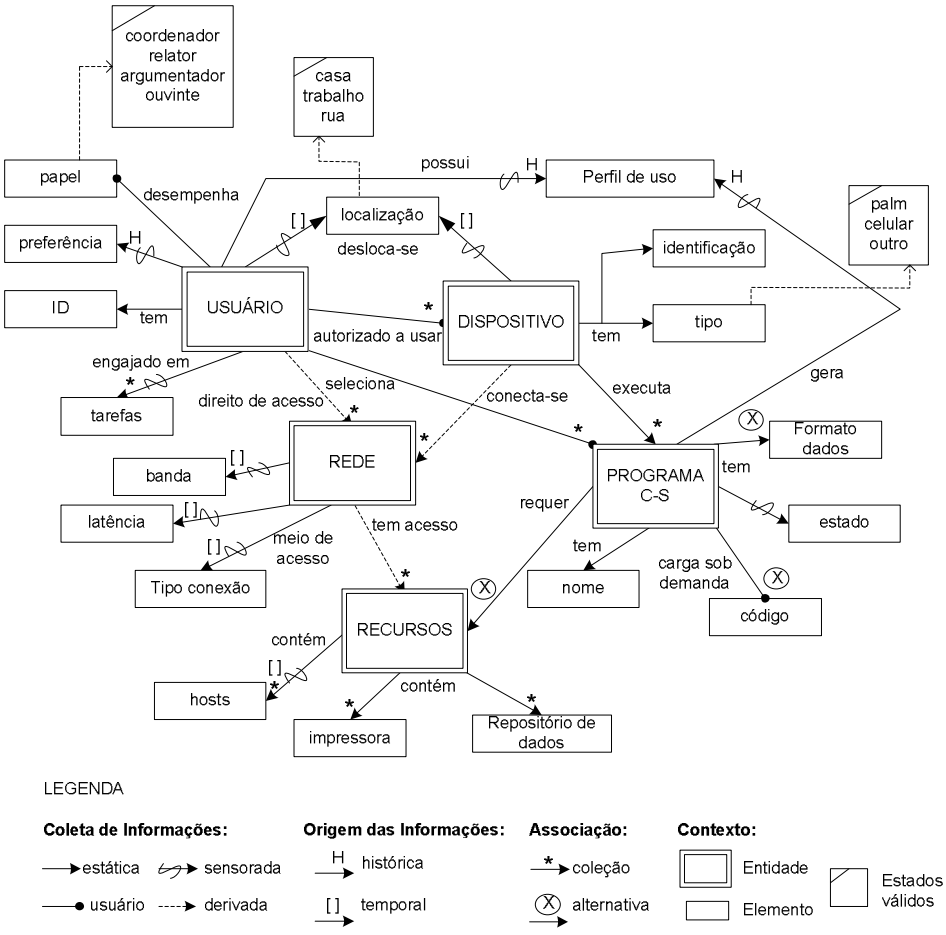


Figura 3.9: Exemplo de Modelagem do Contexto

4 ISAMadapt, PROJETANDO APLICAÇÕES PERVASIVAS

Recentemente, muitas ferramentas para a construção de aplicações móveis têm sido implementadas (ver Capítulo 7). O foco dessas contribuições é em habilitar o acesso aos serviços e informações móveis. Estas não incluem um modelo para aplicações com comportamento *pervasivo*. Assim, essas contribuições foram analisadas com vistas a identificar um conjunto de características que governasse a construção de aplicações móveis adaptativas em um ambiente *pervasivo*.

Em geral, aplicações móveis devem dinamicamente explorar uma entrega de recursos extras, e elegantemente degradar-se quando a disponibilidade do recurso deteriora. A fim de fazer isto, a aplicação deve ser:

a) suficientemente geral para que situações de alteração na disponibilidade de recursos possam ser tratadas, para minimizar o impacto na aplicação;
b) consciente da disponibilidade corrente dos recursos;
c) estruturada de forma que a funcionalidade e o uso dos recursos sejam alterados de acordo com os requisitos da aplicação e a disponibilidade corrente dos recursos.

O sistema de suporte deve, portanto, exportar consciência do ambiente para a aplicação. O que conduz a duas questões: (i) como encontrar as alterações dinâmicas do ambiente; e (ii) como se ajustar aos recursos disponíveis, de forma a manter os objetivos da aplicação. Essas questões nos remetem aos aspectos analisados no ISAMadapt, o qual fornece um modelo, linguagem e suporte em tempo de execução para se construir e executar aplicações *pervasivas*.

4.1 Requisitos e Princípios do ISAMadapt

Na definição do ambiente de programação ISAMadapt, além dos requisitos expostos no Capítulo anterior, alguns princípios de engenharia de software foram considerados para a semântica da solução: a separação de conceitos entre aspectos funcionais da aplicação e aspectos não-funcionais relativos à consciência do contexto; a reusabilidade; e a relação simbiótica do ambiente de programação com o ambiente de execução.

A **separação de conceitos** tem sido reconhecida como uma forma de estruturar programas e gerenciar a complexidade dos sistemas. Paradigmas, como reflexivo (BLAIR, 1998) e orientado a aspectos (KICZALES et al., 1997), têm sido propostos com este objetivo. Um sistema complexo pode ser dividido em subsistemas mais simples e tratáveis. Cada subsistema é relativamente independente dos demais e a solução completa é fornecida por uma colagem desses subsistemas. As partes que compõe o todo são unidades modulares de funções.

Como um sistema *pervasivo* é um sistema complexo, usar o princípio de separação de conceitos **simplifica** e **flexibiliza** o projeto das aplicações. Outra propriedade

buscada pelo ISAMadapt é a **generalidade** da solução, para que esta seja aplicável a vários domínios de aplicações *pervasivas*. A generalização é obtida através da composição e parametrização de componentes básicos, identificados no processo de tornar uma aplicação consciente do contexto em que executa, disponibilizados pelo ambiente ISAMadapt.

A **reusabilidade** também é um princípio desejável para simplificar o projeto de aplicações com comportamento adaptativo.

A necessidade de uma **plataforma comum** é enraizada na considerável e inerente heterogeneidade dos equipamentos do ambiente *pervasivo*. Hoje, aplicações móveis são tipicamente desenvolvidas para classes específicas de dispositivos, como palms, notebooks e celulares, deixando uma versão diferente de uma mesma aplicação para cada equipamento. Além disso, cada aplicação necessita ser distribuída e instalada separadamente para cada classe de dispositivo. Hoje, a sincronização entre os dados das diferentes versões é realizada pelo comando manual do usuário. Com o aumento dos tipos de dispositivos e das aplicações (n-versões) que executam neles, desenvolver aplicações que executam em todas as plataformas tornar-se-á extremamente difícil. Isto conduz a necessidade de uma aplicação única que possa ser implementada em uma faixa de dispositivos, cujo gerenciamento é realizado pelo sistema de suporte e não mais pelo usuário.

A relação simbiótica do ambiente de programação e do ambiente de execução tem sido experimentada com sucesso em várias propostas, desde Unix-C. Outro exemplo é Emerald (JUL et al., 1988), a primeira linguagem com mobilidade forte de objetos que apresentava um sistema operacional correspondente dando suporte às primitivas de mobilidade com vistas a obter gerência e desempenho.

Assim, baseado nesses princípios, e nos requisitos expostos na Seção 2.3, as premissas da solução ISAMadapt são:

a) conceito de contexto definido pela própria aplicação – permite a generalização dos mecanismos de suporte ao contexto;
b) independência entre o reconhecimento do contexto e a aplicação em si – permite o reuso das informações de contexto por outras aplicações, e simplifica o projeto da aplicação;
c) desacoplamento da funcionalidade da aplicação e da adaptação – conduz à flexibilidade e extensibilidade no projeto das aplicações;
d) funcionalidades alternativas para expressar as possibilidades de comportamento em face ao estado do contexto – reduz a complexidade da aplicação ao tratar separadamente cada estado do contexto;
e) reconfiguração dinâmica da aplicação através da seleção automática das alternativas face ao estado corrente do contexto – simplifica o projeto e implementação da aplicação;
f) execução negociada dos mecanismos de adaptação, e decisão de adaptação dirigida pela aplicação – aumenta o desempenho do sistema e satisfação do usuário.

Os principais conceitos, abstrações e componentes do ISAMadapt são detalhados a seguir.

4.2 Estrutura da Aplicação ISAMadapt

Suportar adaptação no nível de aplicação requer que as aplicações tenham sido escritas especificamente para operar em ambientes dinâmicos. Na arquitetura ISAM, a aplicação é projetada com base no HoloParadigma e é organizada como uma coleção de entes (BARBOSA, 2002). A escolha do HoloParadigma foi baseada no fato de que este

atende a alguns dos requisitos da arquitetura ISAM: (i) a comunicação realizada via *blackboard*, permite o desacoplamento temporal e espacial requerido; (ii) a organização estrutural de entes que suportam mobilidade de forma implícita e explícita.

Uma aplicação ISAMadapt é estruturada em quatro tipos de componentes:

a) entes funcionais e adaptativos, que implementam a lógica da aplicação e fazem uso de rotinas adaptativas. Aos entes está associado o modelo de programação e interação da aplicação;
b) adaptadores, que implementam o comportamento das rotinas adaptativas correspondentes ao estado do elemento de contexto corrente. Aos adaptadores está associado o modelo de adaptação adotado;
c) elementos de contexto de interesse da aplicação. A estes está associado o modelo de contexto;
d) políticas de adaptação, que orientam o sistema de execução da adaptação. Às políticas está associado o modelo de colaboração aplicação-sistema de suporte (*middleware*).

A Figura 4.1 fornece uma visão espacial do relacionamento entre entes, contexto, adaptadores e políticas de adaptação, durante o processo de execução da aplicação. Os próximos itens detalham estes componentes e os modelos associados.

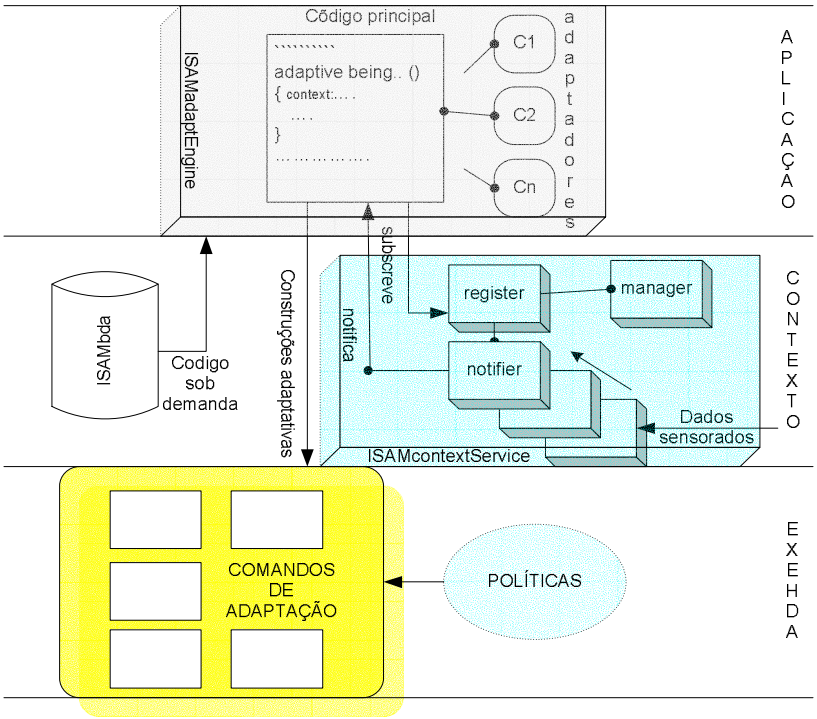


Figura 4.1: Componentes ISAMadapt e seus Relacionamentos

4.2.1 Modelo de Programação e Interação

Sistemas altamente heterogêneos, como o *pervasivo*, com muitas entidades independentes e atividades paralelas requerem modelos de coordenação e programação que apropriadamente permitam expressar a mobilidade. De acordo com a proposta ISAM, a adaptação também deve ser declarada explicitamente.

A mobilidade deve ser expressa por modelos de entidades com características de autonomia, pró-atividade, migração e reação. O modelo de ente do HoloParadigma é similar ao conceito de agentes (PHAM; KARMOUCH, 1998), que apresentam tais

características. Neste sentido, a aplicação modelada com entes tem a vantagem de unir os dois modelos de programação: agentes móveis, e baseado em eventos. Logo, este se torna atrativo para expressar mobilidade e adaptabilidade. Como o HoloParadigma foi concebido com a perspectiva de mobilidade tanto lógica quanto física, sua utilização torna-se natural na proposta ISAM. A Figura 4.2 ilustra duas possibilidades de mobilidade dos entes que são de interesse da arquitetura ISAM: a mobilidade física e lógica (A), e a mobilidade física (B).

Na fase de concepção, o programador dispõe de um ambiente de programação que o orienta na descrição da aplicação e do comportamento adaptativo desta. A modelagem da aplicação é baseada no Holoparadigma e a codificação na correspondente HoloLinguagem (www.inf.ufrgs.br/~holo) acrescida das abstrações/comandos ISAMadapt.

Em Holo, 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, sua história. Um ente pode ser composto de outro ente (Figura 4.2). 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 ente, torna-se seu filho.

A comunicação e sincronização, entre os entes, é realizada através da **história**, que funciona como uma memória distribuída logicamente compartilhada. O modelo Holo define que um ente somente tem acesso a sua própria história e a história do seu ente pai. Para a comunicação com outro ente, o ente pode se deslocar para dentro de outro ente, 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. Como a comunicação entre entes é modelada através do paradigma de Espaço de tuplas-objetos, esta mantém a uniformidade com o modelo de comunicação projetado para a arquitetura ISAM.

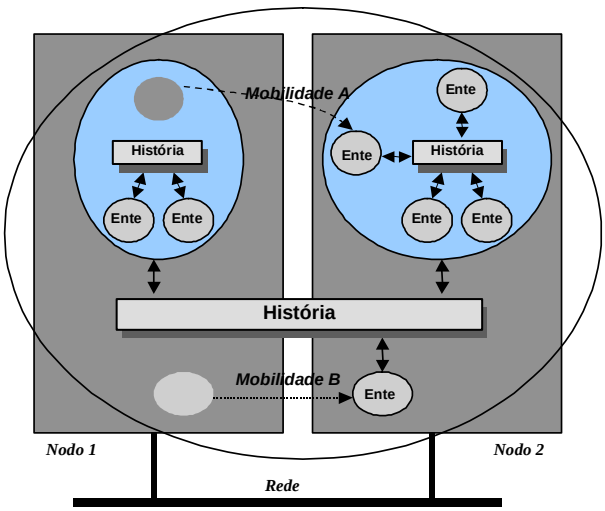


Figura 4.2: Mobilidade no HoloParadigma

4.2.2 Modelo de Adaptação ISAMadapt

Para definir o processo de adaptação, deve-se responder às perguntas: “que entidades se adaptam?”, “quem executa a adaptação?”, “onde ocorre a adaptação?”,

"quando ocorre a adaptação?", "como ocorre a adaptação?". Os itens abaixo respondem estas questões sob a ótica do ISAMadapt.

(a) Que entidades da aplicação se adaptam?

ISAM procura um conceito flexível de adaptação, o qual está relacionado com o contexto da aplicação. Assim, cada ente da aplicação ISAMadapt pode determinar a que elemento(s) de contexto(s) é sensível, e quais de seus **métodos** têm um comportamento adaptativo. Outros entes da aplicação, os **entes adaptativos**, diferem destes por terem todo o seu comportamento sensível ao contexto. Aos entes são associados **adaptadores** que tratam das variações contextuais.

(b) onde ocorre a adaptação?

A resposta depende de onde estiver localizado o ente da aplicação que expressa um comportamento adaptativo: no cliente móvel, no servidor ou em um nodo intermediário. Em Odyssey (NOBLE, 2000), a aplicação no cliente móvel reage às trocas na largura de banda; em Mobiware (ANGIN et al., 1998) cliente e servidor se adaptam; enquanto que em Timely (BHARGHAVAN; GUPTA, 1997), agentes procuradores reagem em nome do cliente. ISAMadapt adota uma abordagem flexível, onde ambos, **dispositivo móvel e estático**, podem conter entes que estão associados a adaptadores.

(c) quando ocorre a adaptação?

Esta questão está associada à propriedade de **agilidade** do sistema (NOBLE, 2000) – o sistema deve reagir à alteração tão rápido quanto possível. A noção de agilidade é complexa, pois diferentes aplicações podem ter diferentes sensibilidades a diferentes trocas nos elementos do contexto. Por exemplo, uma aplicação pode ser mais sensível à banda que ao nível de energia. Outra fonte de dificuldade é as diferentes origens das alterações na disponibilidade dos recursos: variação na oferta devido à mobilidade, ou devido ao aumento na demanda de recursos compartilhados. A agilidade ideal é difícil de ser alcançada, pois é um equilíbrio entre a responsividade e a intrusividade das adaptações que a aplicação pode sofrer num período de tempo.

A adaptação ISAMadapt ocorre em dois tempos: ***load-time*** e ***run-time*****.** O processo de adaptação é controlado pela `ISAMadaptEngine`, componente integrante do ambiente de execução da linguagem. Como ISAMadapt usa carga dinâmica de código para os entes da aplicação com adaptadores associados, a `ISAMadaptEngine` pode identificar o contexto atual e carregar o código já adaptado ao ambiente corrente. Os códigos da aplicação são armazenados na `ISAMbda`, uma base de código *pervasiva* gerenciada pelo ambiente de execução. Conforme a execução da aplicação progride, novas adaptações podem ocorrer.

A agilidade com que ocorre a adaptação em *run-time* é determinada pela aplicação ISAMadapt ao definir as possibilidades de estados de um elemento do contexto ao qual é capaz de se adaptar. Cada aplicação pode definir um número de possibilidades diferentes, dependendo de seus objetivos, para o mesmo elemento de contexto. Desta forma, as aplicações podem ter valores de agilidade diferentes.

Outro aspecto a determinar é quando o ente em execução pode ser interrompido para se reconfigurar. Por questões de simplificação do gerenciamento, o ente não recebe diretamente a notificação de alteração do elemento do contexto; é a `ISAMadaptEngine` que recebe a notificação do `ISAMcontextService`. Esta mantém uma tabela de configuração da aplicação relacionando entes, adaptadores e estado dos elementos de contexto. Com base nessas informações e no contexto corrente,

a `ISAMadaptEngine` seleciona o código do método a ser carregado. Na próxima vez que o ente em execução chamar o método, o novo comportamento (adaptado) será executado. Desta forma, a adaptação somente ocorre no início de um método. Soluções como a cada iteração de um laço foram avaliadas como dispendiosas, tanto em tempo quanto em gerenciamento. Uma solução intermediária, planejada para avaliação futura, é considerar blocos de adaptação. No código da aplicação seriam inseridos marcadores de início e fim de bloco, quando entra neste bloco a aplicação poderia se adaptar.

(d) quem executa a adaptação?

Noble (2000), argumenta que a adaptação deve ser consciente da aplicação, e que esta é responsabilidade do sistema e da própria aplicação. O sistema Odyssey, onde a aplicação decide o grau de fidelidade em relação aos dados originais face à alteração na largura de banda do ambiente móvel, demonstra a viabilidade e adequação de sua tese ao ambiente móvel. Seguindo esta linha de pensamento, ISAM adota um **modelo colaborativo** entre a aplicação ISAMadapt e o *middleware* EXEHDA (Seção 3.4.2). Devido ao sistema ISAM ser orientado a aplicações de propósito-geral, diferente de Odyssey, o problema é determinar o grau de cooperação requerido entre aplicação e sistema. Para resolver esta questão, ISAMadapt usa o conceito de políticas de adaptação, definidas pela aplicação e seguidas pelo EXEHDA na execução dos comandos de adaptação.

(v) como ocorre a adaptação?

A arquitetura ISAM oferece a característica de adaptação numa combinação da **forma implícita** com a **explícita.** O modelo de programação, usando o conceito de adaptadores e comandos de adaptação, facilita a implementação de uma aplicação adaptativa. Argumenta-se que os componentes da aplicação devem ter capacidades de (re)configuração automática a fim de manter sua reusabilidade e manutenibilidade. Outro novo aspecto é a incorporação da adaptação ao contexto às construções da Hololinguagem: migração e clonagem. Separados do conceito de ente propriamente dito, o Serviço de Reconhecimento de Contexto e os adaptadores permitem o reuso do código e facilitam a manutenção da aplicação.

Os **adaptadores** separam a especificação do comportamento adaptativo potencial da funcionalidade da aplicação em si. O **Serviço de Reconhecimento de Contexto** opera independentemente da aplicação, e esta se reconfigura de forma automática dependendo do contexto atual e da decisão do ambiente de execução EXEHDA. Ambos, ISAMadapt e EXEHDA, determinam quando a adaptação deve ser executada e como o código da aplicação deve ser modificado (isto é, qual de seus adaptadores é escolhido) baseado nas tabelas de configuração da aplicação. A sintaxe e semântica da adaptação ISAMadapt é detalhada no item 4.3.

Outro aspecto da adaptação ISAM é que esta pode ser **interna** – determinada pelo código dos adaptadores da aplicação, selecionados em função do contexto de execução, e **externa** – determinada pelo gerenciamento do ambiente de execução, com vistas a maximizar a qualidade de serviço prestado às aplicações (YAMIN, 2004). Do ponto de vista da aplicação, o *middleware* EXEHDA executa funções como criação e migração dos componentes da aplicação, e colabora no processo de adaptação da aplicação. É também responsável pela ativação do `ISAMcontextService`, componente responsável por obter informações do estado do ambiente corrente. O sistema EXEHDA é o responsável, em última instância, pela realização da adaptação, pois as decisões para adaptação não devem ser somente baseadas em informações locais, mas também com base em informações de outras aplicações.

É importante notar que, neste momento, ISAM aborda somente a questão da adaptação à variação na disponibilidade de recursos, considerando que sempre existem os recursos mínimos para a execução da aplicação. A intenção é fazer o melhor uso do recurso disponível. Futuramente, um gerenciamento de recurso pode usar o modelo de aplicação para dinamicamente realocar, reescalonar e reestruturar as aplicações e seus recursos.

4.3 Abstrações do ISAMadapt - Sintaxe e Semântica

Um sistema de adaptação dinâmica, em geral, é estruturado em três atividades: monitoramento, interpretação e reconfiguração. No ISAM, o sistema de suporte deve registrar o interesse da aplicação em uma informação de contexto específica; coletar informação relevante para a aplicação e interpretar a informação de contexto transformando-a em uma informação de mais alto nível, a qual permitirá a aplicação selecionar entre alternativas de adaptação (`adapters`).

Considerando esta funcionalidade, as abstrações disponibilizadas na linguagem são relativas à consciência de contexto e à adaptação dinâmica, e compreende: contexto, comportamentos alternativos, comandos e políticas de adaptação. Essas abstrações são implementadas em duas visões: tempo de programação e tempo de execução. Portanto, o processo de adaptação é realizado em duas fases: programação e execução (Figura 4.3).

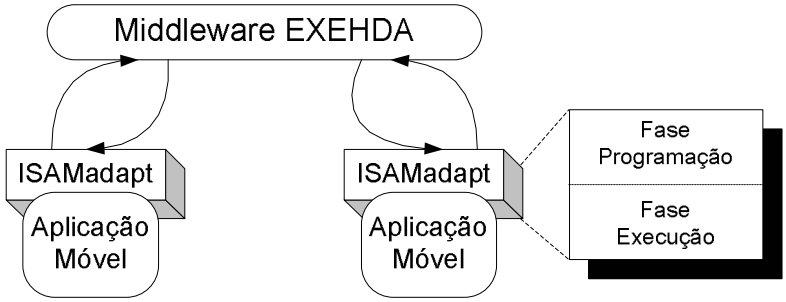


Figura 4.3: Estrutura Organizacional do ISAMadapt

4.3.1 Fase de Programação

Abstrações orientadas à adaptação dinâmica são introduzidas para permitir expressar o comportamento adaptativo dos componentes da aplicação e dirigir sua execução. Nesta fase, o programador ISAMadapt codifica os vários aspectos da adaptação: define os **elementos de contexto** de interesse da aplicação, codifica **comportamentos alternativos** ajustados às condições ambientais e seleciona as **políticas de adaptação** que orientam o sistema de execução. Esta organização do projeto da aplicação conforma-se com uma metodologia (ainda embrionária) para a criação de aplicações conscientes do contexto (REIS et al., 2002). Uma metodologia de projeto é mais efetiva se também estiver embutida no ambiente de desenvolvimento da aplicação (linguagem e ferramentas).

4.3.1.1 Declarando os Elementos de Contexto

Características do mundo tornam-se contexto através de seu uso (WINOGRAD, 2001). Desta forma, o contexto focaliza somente os aspectos que são relevantes em uma situação particular. Cada aplicação pode selecionar elementos de contexto diferentes, ou

selecionar os mesmos elementos de contexto, porém com interpretações diferentes. Por exemplo, uma aplicação pode ser sensível à rede e considerar somente o aspecto conexão/desconexão, enquanto que outra aplicação pode considerar a largura de banda, identificada como alta, média, baixa; ainda, outra aplicação pode considerar a largura de banda nos estados: altamente conectado, fracamente conectado, desconectado.

Assim, cada aplicação deve definir o que significa contexto para si. O ISAMadapt fornece a generalidade que permite a especialidade do contexto para cada aplicação. Isto é realizado através da etapa de *definição dos elementos de contexto*. No ISAMadapt, usa-se o menu `Context` do `ISAMadapt IDE` para especializar os elementos de contexto da aplicação. Variáveis de ambiente e a declaração `context,` usados no código, também declaram o contexto da aplicação. Estes elementos são descritos a seguir.

ISAMadapt IDE: menu *Context*

Com o princípio de separação de conceitos, a especificação do contexto da aplicação não está embutida no código da aplicação, mas é um componente à parte no projeto da aplicação. No ambiente de desenvolvimento, o menu `Context` define e configura a entidade de interesse da aplicação, ao qual esta poderá se adaptar. Este funciona como um modelo (*template*) que orienta o programador na definição dos atributos dos elementos de contexto necessários para personalizar a execução do `ISAMcontextService` para a aplicação.

Informações coletadas no menu `Context` são armazenadas no arquivo de descritores de contexto, no formato XML (*eXtended Markup Language*). Estas informações são usadas pelo compilador para gerar chamadas de rotinas que reconfiguram o `ISAMcontextService` para atender às necessidades da aplicação. Um esboço das informações deste menu é apresentado na Figura 4.4.

Dois tipos de elementos de contexto são definidos:

(a) nativos – pré-definidos no ambiente, são os de uso comum, como localização, banda, tipo do nodo e número de processadores disponíveis;

(b) da aplicação – relacionam-se a elementos que são da aplicação em específico, como por exemplo: grau de interação entre métodos, e eventos associados ao uso de determinado banco de dados.

Devido a questões de configuração do Serviço de Contexto, relativa a forma de monitoração e tradução, definem-se dois tipos de contexto: contexto direto e contexto especializado. Contexto direto é aquele que tem somente dois estados: disponível, não disponível, e fazem uso direto dos dados sensorados. Por exemplo, processador disponível. Este tipo de contexto também está relacionado ao conceito de descoberta de recursos, uma função fornecida pelo `ISAMcontextService`. Contexto especializado é aquele que reflete a visão particular da aplicação, onde os dados sensorados passam por um tratamento mais elaborado a fim de atendê-la (ver Apêndice B).

Código Fonte: Variáveis de Ambiente

O programador pode fazer uso direto das variáveis de ambiente, mantidas atualizadas pelo ambiente de execução, nos comandos da linguagem. As variáveis inicialmente previstas são: *userLocation* – localização corrente do usuário; *ISAMbda* – identificação da base de dados mais próxima, *ISAMcontextService* – identificação do

serviço mais próximo, e variáveis integrantes do perfil do usuário: *myHost, myActivities, myPreferences*.

Por exemplo,

```
if  (userLocation = myHome) {
     install applic1 at env:myHost [start];
}
```

Este comando orienta o EXEHDA para instalar a aplicação *applic1* no dispositivo do usuário, em sua casa.

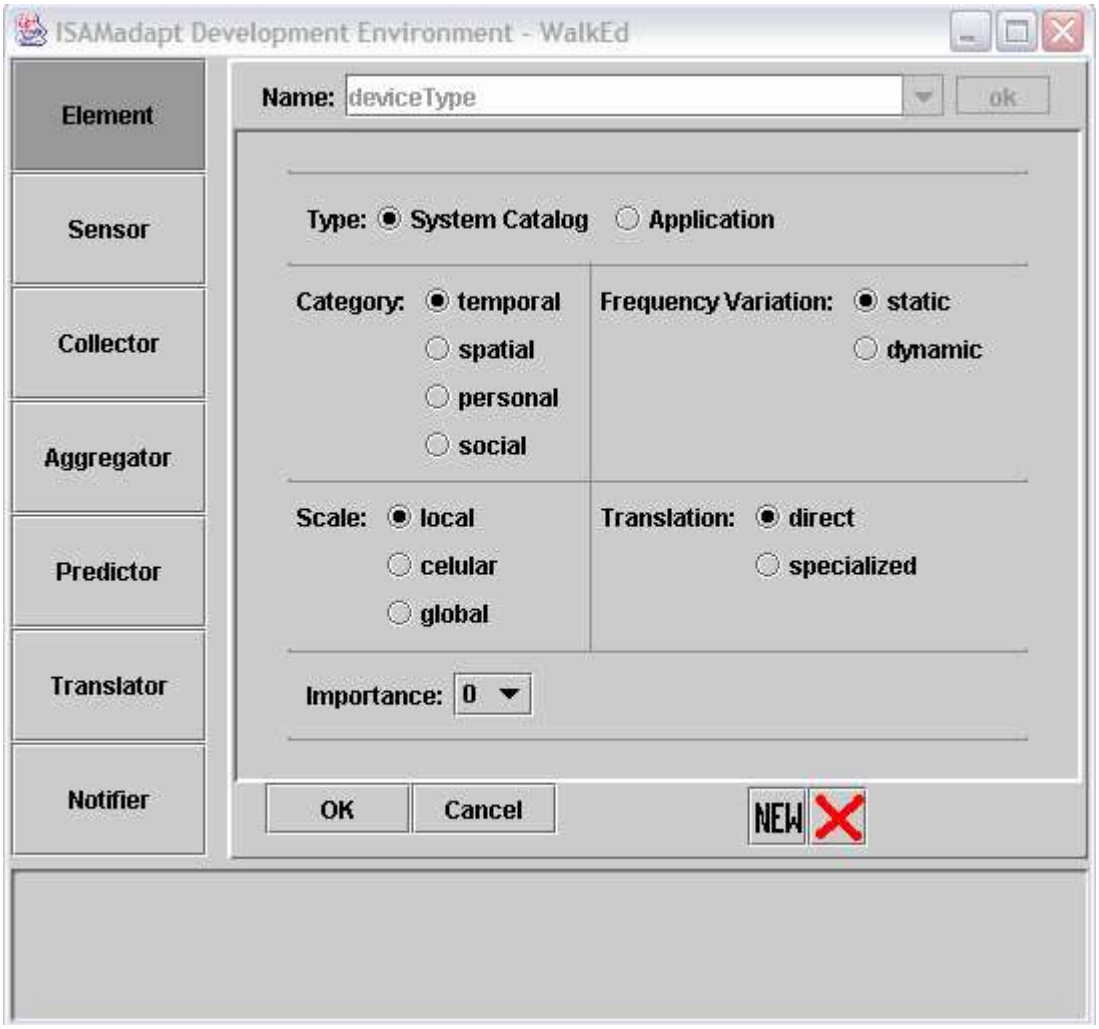


Figura 4.4: Esboço Menu `Context` do Ambiente de Desenvolvimento ISAMadapt

Código Fonte: Comando context

Usa-se o comando ***`context`*** para declarar um elemento de contexto a cujo estado o ente da aplicação se adapta. Este comando de declaração também está associado à declaração de métodos e entes adaptativos (Seção 4.3.1.2). A forma geral do comando de declaração de contexto é

```
context <nome_elemento_ctx> ;
```

a qual expressa o nome do elemento do contexto previamente definido pelo sistema ou criado pelo programador. Por exemplo, *`context  processor_idle`*.

A aplicação pode especificar o(s) estado(s) do elemento de contexto ao qual é sensível. Neste caso a sintaxe é

```
context <nome_elemento_ctx> :: (lista_estados);
```

Por exemplo, `context network::(connected, disconnected)` indica que a aplicação se adapta ao estado conectado/disconectado da rede.

Com o objetivo de documentação, o programador pode usar a diretiva:

```
//@context: <nome_elemento_Contexto> :: <lista_estados>
```

A declaração do contexto, através do comando `context`, permite que o elemento de contexto seja usado em condições associadas a comandos de seleção e repetição. Por exemplo:

```
if (~ processor_idle)  { disconnect; }
  // desconecta se não existe processador disponível

{  context display :: (pda, notebook)
   if ( display = pda)  .....
}
```

Além disso, a aplicação pode controlar o recebimento de notificação de alteração no estado do contexto através das funções `suspendNotify` e `resumeNotify`, que suspendem e retornam o recebimento de notificação respectivamente. Aplicação também pode consultar o estado do elemento de contexto através das funções `getContext` – retorna o valor contido no servidor, e `checkContext` – força uma monitoração do contexto antes de retornar seu estado.

4.3.1.2 Expressando Comportamento Adaptativo

Em geral, as estratégias adaptativas utilizadas nas aplicações móveis podem ser caracterizadas por atributos ou requisitos operacionais. Atributos alteram-se com a execução, como o formato dos dados de imagem alterado para uma descrição textual correspondente. Requisitos operacionais são os recursos necessários para a execução. Exemplo: modo de transmissão, recursos como impressora e banco de dados, requisitos de consistência.

Essas estratégias também variam na localização do mecanismo de adaptação:

(a) embutidos na aplicação - mecanismos específicos são construídos pelo programador dependendo do tipo da aplicação e de sua intenção;

(b) não embutidos na aplicação - mecanismos de adaptação podem ser específicos ou genéricos se fornecidos pelo sistema. Em geral, a aplicação usa esses mecanismos atualizando os parâmetros de sua interface.

Os mecanismos ISAMadapt devem ser genéricos o suficiente para permitir expressar as estratégias de adaptação observadas na maioria das aplicações móveis.

Para introduzir código de adaptação, têm-se três alternativas:

a) imperativa, os comandos são expressos no código da aplicação;
b) reflexiva, o código é confinado em metaobjetos associados aos objetos da aplicação;
c) diretiva, instruções para o ambiente configurar os mecanismos de adaptação que este disponibiliza. Considerando os aspectos positivos de cada abordagem, optou-se por introduzir um mecanismo que as combinasse.

Além disso, foram identificados três tipos de adaptação (AUGUSTIN, 2002b): (i) paramétrica – altera somente o conteúdo de variáveis; (ii) alternativa – altera o código, mas mantém a funcionalidade; (iii) funcional – altera a funcionalidade em resposta à

nova situação de uso. Neste momento, ISAMadapt trata somente as duas primeiras, aplicações *situation-aware* (iii) não foram ainda consideradas.

A adaptação ISAM requer a existência de múltiplos caminhos de execução da aplicação, ou configurações alternativas, as quais exibem diferentes níveis de utilização de recursos. Esses múltiplos níveis fornecem flexibilidade em tempo de execução, permitindo escolher entre as configurações alternativas a mais adequada à situação corrente. As configurações podem ser construídas de diferentes códigos-adaptadores da aplicação (adaptação de código) ou de um mesmo código-adaptador exibindo comportamento diferente sob a influência de alguns parâmetros de controle (adaptação paramétrica).

Embora a transição entre diferentes configurações possa ser realizada no nível da aplicação, considera-se que esta é mais controlada no nível do sistema, sem o envolvimento direto do programador. Entretanto, isto requer algum caminho dependente da aplicação para enumerar as diferentes configurações tanto quanto um mecanismo para transição entre as diversas alternativas. Encontram-se esses requisitos no conceito de **adaptadores**, introduzidos no contexto da linguagem. Este fornece o suporte para expressar a disponibilidade de múltiplos caminhos de execução, e o ISAMadapt *runtime* fornece os meios para que o progresso da aplicação seja monitorado e influenciado pelo contexto (chaveamento de diferentes caminhos).

No ISAMadapt são disponibilizadas várias formas de tornar o código consciente do contexto. Está a cargo do projetista da aplicação decidir qual a forma mais convenite para cada caso. Para expressar o comportamento adaptativo são fornecidas construções (i) baseadas em contexto e (ii) baseadas em comandos de adaptação.

(i) Construções Contextualizadas

Para tratar com elementos de contexto pode-se usar o comando **`onContext`**, associado à declaração de `context` (Seção anterior). Este comando cria um bloco de comandos que serão executados quando o contexto tornar-se disponível. Este comando apresenta uma semântica assíncrona, a qual permite ao ente continuar executando suas tarefas enquanto comandos podem ficar à espera do contexto que necessitam. O comando ficará bloqueado na `ISAMadaptEngine` até receber a notificação do `ISAMcontextService`[9], enquanto o restante do código do ente segue sua execução. Para evitar a postergação infinita do comando bloqueado, políticas de adaptação global podem definir um valor de temporização (*timeout*) para controle via middleware EXEHDA.

O próximo exemplo codifica a impressão de um arquivo (`file1`) quando uma impressora colorida estiver disponível (na área de acesso do cliente móvel).

```
context   colorPrinter;
......
onContext colorPrinter { clone (printer(file1),#impressora);  }
```

Neste exemplo, o comando ficará bloqueado até receber a notificação de impressora disponível, porém o ente continuará sua execução no próximo comando.

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

```
context  new_mail;
....
```

[9] Por questões de eficiência, será verificado o contexto antes de agendar a tarefa na `ISAMadaptEngine`.

```
sync   onContext new_mail {
     checkMail ();
}
```

Neste, o método para verificar o correio eletrônico somente é executado após a chegada de um novo `e-mail`.

Para tratar com contextos especializados, cujos estados são determinados e nomeados pela aplicação, ao comando `onContext` é acrescido a definição do estado do elemento de contexto com o qualificador ::. A sintaxe do comando é:

```
onContext <nome_elemento-contexto>  :: <estado>
```

Por exemplo,

```
context  accessNetwork::(free, firewall);
....
while (true) {
     onContext  accessNetwork::free  {  clone(worker,  #worker_id);
}
```

Usando o `Context menu` da `ISAMadaptIDE`, o programador configura o elemento de contexto `accessNetwork` para que seus estados sejam traduzidos como "firewall" ou "free".

Para codificar a adaptação paramétrica, pode-se utilizar uma combinação dos comandos de contexto com comandos de seleção de forma a atribuir valores às variáveis-parâmetros dependentes do contexto. Por exemplo, a chamada de um procedimento cuja porta de conexão irá variar dependendo do contexto tipo de acesso à rede, `connection (portal.inf.ufrgs.br)`, tem a sintaxe para sua declaração:

```
connection (host)
  {   context accessNetwork:: (firewall, free);
      int porta;
      …
      if  (accessNetwork == firewall )
            { porta := 80; )
     else if (accessNetwork == free )
               { porta := 1099; }
           else {  return (msg); }
    openConnection (host, porta);
    …
}
```

O comando `if` ficará bloqueado até que a tupla `<accessNetwork, firewall>`, `<accessNetwork, free>` ou `<accessNetwork, unkown>` esteja disponível no espaço de objetos da `ISAMadaptEngine`. Esta tupla reativa (acorda o comando) é gerada pelo Serviço de Contexto em resposta à subscrição de notificação do tipo de acesso ao `host`.

Para a adaptação de código, onde a definição funcional da aplicação não é alterada, mas define algoritmos alternativos com variados níveis de recursos, existem duas possibilidades: (1) ente adaptativo; (2) método adaptativo.

Ente Adaptativo

Ente adaptativo é aquele cujo código é determinado pelo contexto. Em geral, associado ao código de carga, na instanciação do ente. Por exemplo, um ente

`interface` cujo código é dependente do tipo de dispositivo que dispara sua criação. A sintaxe de declaração do ente é:

```
adaptive being <nomeEnte> (<parâmetros>)
  context <nomeElementoContexto>::(<lista_estados_possíveis>)
{ // métodos compartilhados   }
```

onde, a lista de estados possíveis do elemento de contexto é opcional.

Por exemplo, a declaração

```
adaptive being interface
  context   deviceType :: (pda, desktop);
{    }
```

diz para o sistema que ao criar o ente `interface` carregar seu código correspondente ao estado do elemento de contexto `deviceType`.

A implementação dos códigos alternativos para o ente é através de **adaptadores** (`adapters`), os quais funcionam como *containers* que modelam comportamento aos recursos disponíveis. Desta forma, adaptadores são construídos como pares do estado do ambiente em execução, e apropriados procedimentos da aplicação para o estado. Fica a cargo da `ISAMadaptEngine` selecionar o código adequado para *dispatch/load* (Seção 5.4).

Adaptadores de entes são codificados da mesma forma que entes, somente com uma sintaxe de identificação diferente, e são armazenados em arquivos separados com a extensão `adp`. A sintaxe para definir adaptadores para os entes é:

```
// ... file  *.adp
//@context: <contexto> :: ( <lista_estados_possiveis>)
//@description: descrição para documentação

adapter being <nome_do_ente>::<estado_ctx> (<parâmetros>)
{ // código Holo + ISAMadapt
.....
}
```

Por exemplo,

```
//@context: display :: (bw, color)
// file: interface_bw.adp
//
adapter being interface::bw ( )
{ // codificação para o estado de interface preto e branco
...
}
```

declara um adaptador para o ente interface com estado do contexto display preto-branco. O elemento de contexto tem somente a função de documentação no código. A não amarração deste ao adaptador permite a reutilização deste adaptador em outra situação ou aplicação. A amarração válida para fins de tradução é a realizada através do `Adapter menu` da `ISAMadapt IDE`.

Método Adaptativo

Método adaptativo é um método cuja funcionalidade é adaptativa ao contexto, comporta-se como uma função genérica. Estes são identificados no código pelo atributo `adaptive` acrescido ao nome do método chamado. 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;
```

diz que `outResult` será implementada através de adaptadores (`adapter`) para cada estado do contexto `deviceType`. Por exemplo, se o dispositivo em uso é um Palmtop, a saída pode ser um subconjunto dos dados; se for um celular, pode ser somente o totalizador dos dados; se for um *notebook*, pode ser a relação completa dos dados.

A chamada de método segue o padrão Holo:

```
<nome_do_método> (<parâmetros>);
```

A sintaxe para codificação de **adaptadores** de métodos é:

```
// ... file  *.adp
//@context: <contexto>
//@description: descrição para documentação
adapter      method      <nome_ente>.<nome_método>::<estado_ctx>
(parameters)
{ ... code  ....}
```

Por exemplo,

```
// ... file  *.adp
//@context: deviceType::cdcDevice
//
adapter method interface.outResult::cdcDevice(what)
{
 // código de dados de saída resumidos ou convertidos para
 //formatos dos  PDAs do tipo CDC
 ...
}

// ... file  *.adp
//@context: deviceType::laptop
//
adapter method  interface.outResult::laptop(what)
{ // código de dados de saída completos usando
      // toda tela dos  laptops
        ...
}
```

Cada adaptador declara o estado corrente no qual o algoritmo codificado usa plenamente os recursos disponíveis. Esta estruturação permite a inserção, remoção, substituição de novos estados do elemento de contexto, somente criando ou alterando os adaptadores, sem alterar o código do ente original.

ISAMadapt IDE: menu *Adapters*

O código dos adaptadores é associado ao estado do elemento de contexto, ente e método através do menu `Adapter` do ambiente de desenvolvimento ISAMadapt. A Figura 4.5 esboça um layout das informações contidas na interface `Adapters`. O programador também seleciona qual o código *default* que será carregado quando informações de contexto não estiverem disponíveis. As informações obtidas são gravadas num arquivo XML, que integrará o processo de tradução. A máquina de execução `ISAMadaptEngine` selecionará o código do adaptador adequado para execução, quando notificada da troca de estado do contexto, ou em tempo de inicialização da aplicação, e fará uma carga dinâmica do código a partir da `ISAMbda`.

(ii) Comandos de Adaptação

ISAMadapt fornece comandos de adaptação para modelar estratégias de adaptação comuns nas aplicações móveis. Estes podem ser associados a outros comandos, como seleção (`if`) e repetição (`while`). A implementação dos comandos é gerenciada pelo *middleware* EXEHDA, com consulta às políticas de adaptação (Seção 4.3.1.3) definidas pelo programador da aplicação, e está detalhada em (YAMIN, 2004). Os comandos fornecidos são: `move, clone, prefetch, disconnect, reconnect, install, reschedule, discovery.`

move

Este comando fornece tanto a migração (lógica) do ente para dentro de outro quanto indicativo ao escalonador do sistema para realizar a mobilidade física do ente. Esta operação é realizada, em geral por motivos de comunicação.

Sintaxe:

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

Por exemplo, `move to parent` indica que o ente é movido para o ente hierarquicamente superior (seu pai). Neste caso, não é garantido que será realizada uma mobilidade física do ente, pois ela está a critério do *middleware* de execução. Para forçar uma mobilidade física, deve-se usar o comando `move closerTo parent.`

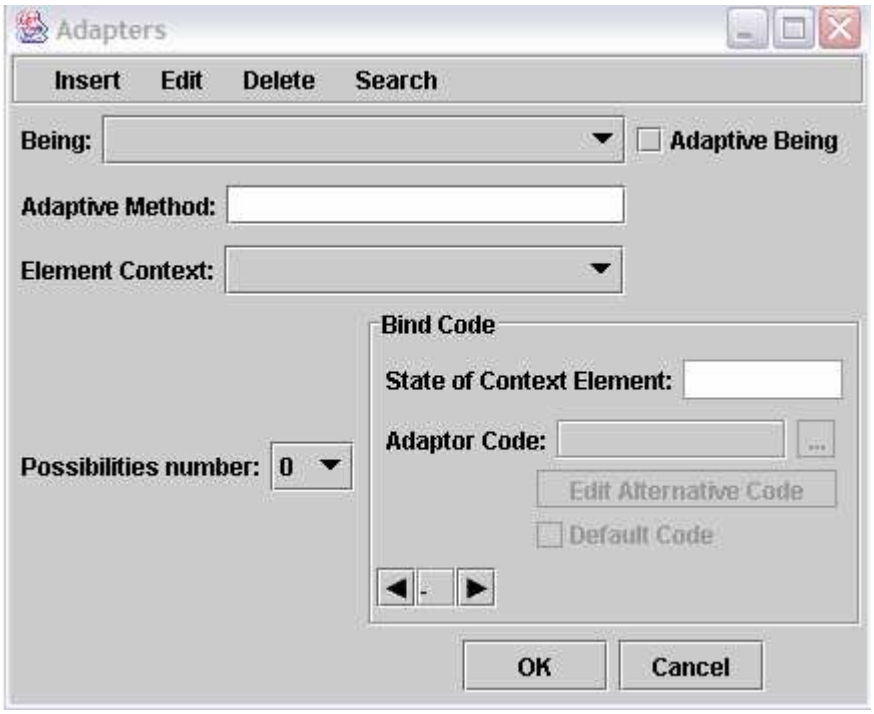


Figura 4.5: Esboço do Menu `Adapters` do `ISAMadapt IDE`

O comando `move` expressa um comportamento pró-ativo, o próprio ente é quem sempre migra para dentro de outro ente (mobilidade lógica) ou para próximo a um recurso (mobilidade física). O comando `move closerTo resource:BD1` indica uma mobilidade física do ente para o dispositivo mais próximo de onde está localizado o recurso nomeado `BD1`.

A semântica original deste comando Holo foi alterada para expressar um comportamento sensível ao contexto e sempre pró-ativo. Os entes são gerenciados pelo

sistema para seguirem o usuário e para estarem sempre o mais próximo possível dos recursos que necessitam (coesão física). O controle físico da migração está a cargo do ambiente de execução EXEHDA, com colaboração da aplicação através da definição de políticas para este comando, ou dos qualificadores inseridos neste comando. Além disso, este comando pode estar associado ao comando de descoberta de recursos (ver comando `discovery`).

clone

O comando `clone` fornece a criação e replicação de entes.

Sintaxe:

```
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 ao gerenciamento físico 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.

O nodo onde o ente é fisicamente criado é determinado pelo EXEHDA com orientações indicadas nas políticas para escalonamento e clonagem, ou expressas neste comando. Se nada for especificado, o ente é criado no dispositivo que solicitou sua criação (*local host*). Este comando pode também ser associado ao comando de descoberta de recursos (ver comando `discovery`).

prefetch

Este comando realiza a busca antecipada de arquivos.

Sintaxe:

```
prefetch <fileName>(, <fileName>)* to (device |<String>) ;
```

Os arquivos são procurados nas bases de armazenamento *pervasivo*, `ISAMbda`, mais próximas, e armazenados no nodo corrente (`device`) ou no AVU do usuário. Este comando, em geral, está associado a um procedimento anterior a emissão de um comando `disconnect`, no qual as informações necessárias para a continuidade da aplicação são armazenadas localmente.

push

Este comando realiza o envio de dados para o usuário a partir do servidor (disseminação de dados). A semântica do comando é contrária à solicitação por demanda do usuário. Sintaxe:

```
push (content) to resource:logicName ;
push (content) to useId ;
```

disconnect

Este comando sinaliza a desconexão lógica da rede, sempre voluntária[10]. A desconexão física da rede é realizada no ISAM 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. O processo de desconexão/reconexão é definido e implementado pelo EXEHDA (YAMIN, 2004).

Sintaxe:

```
disconnect;
```

Políticas de adaptação (Seção 4.3.1.3) poderão definir a estratégia de comunicação postergada a ser usada quando desconectado da rede.

reconnect

Indica reconexão à rede, sincroniza a comunicação mantida no *proxy*.

Sintaxe:

```
reconnect;
```

Indica a necessidade de conexão física à rede. Esta é realizada pelo EXEHDA, a menos que outra aplicação, executando no dispositivo, já o tiver feito.

install

Permite disponibilizar uma aplicação no ambiente virtual do usuário, para uso do usuário, em qualquer lugar, com qualquer dispositivo. Quando da execução, o código da aplicação será buscado na `ISAMbda` mais próxima, e a aplicação é instanciada remotamente pelo EXEHDA.

Sintaxe:

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

onde, `idUser` define a identificação do usuário, cujo AVU 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.

reschedule

Permite ao programador certo controle sobre o escalonamento físico. Este aspecto é importante em aplicações do domínio *Grid Computing*. Sob certas condições do ambiente, definidas pelo programador, a aplicação pode forçar um rescalonamento para melhorar a eficiência.

Sintaxe:

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

onde, `rule` indica a regra a ser usada a partir da emissão do comando. As regras são pré-definidas na política associada ao ente ou recurso especificado. Se nada for especificado, é usado o algoritmo de escalonamento *default* implementado pelo TIPS (REAL, 2003).

Por exemplo,

```
onContext processor_idle { reschedule all; }
```

[10] O ISAM não aborda, ainda, o problema da desconexão advinda de fatores externos, como bloqueio da comunicação devido ao meio.

indicará ao sistema para fazer novo escalonamento de todos os entes quando processadores tornarem-se disponíveis. Se nada for declarado, o comando atua sobre o ente que o chamou. Se este é um ente composto, há um efeito cascata de aplicação do comando. Além disso, a implementação do comando obedece às políticas específicas definidas para o ente (ver Seção 4.3.1.3).

Outro exemplo, o comando

```
onContext centerGeographic
  { reschedule all (policy:centerGeometric); }
```

considera um rearranjo da alocação dos entes da aplicação se há alteração no centro geográfico de acesso, originado pelo deslocamento dos usuários.

discovery

Permite solicitar ao `ISAMcontextService` uma descoberta de recurso ou serviço. 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. Um protocolo de definição de recursos e serviços será definido pela implementação do componente Discoverer do ISAMcontextService (FONTOURA, 2003). Este indica, entre outros elementos, a dimensão da procura (local, celular ou global), os atributos do serviço/recurso (nome, palavras-chaves), *timeout*.

Sintaxe*:*

```
discovery var (<def_file>)  { <commandos> }
```

Por exemplo:

```
discovery  BD ("ava:/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 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,` integra o código da aplicação, será resolvido e esta informação é armazenada no Ambiente Virtual da Aplicação (AVA). Após retorno da resposta, esta é gravada na história do ente, e entes da aplicação serão criados nos nodos que contém o recurso.

4.3.1.3 Definindo Políticas de Adaptação

Políticas implementam o mecanismo de colaboração entre o sistema e a aplicação, e estão associadas à implementação dos comandos de adaptação e gerenciamento destes pelo EXEHDA. Referem-se às orientações da aplicação para a tomada de decisão do sistema, que controla o comportamento geral da aplicação. A principal vantagem desta abordagem é o equilíbrio entre transparência (uso de políticas *default*) e consciência da aplicação. A adaptação automática somente atinge objetivos de propósito-geral, e não considera as necessidades peculiares de cada aplicação. Quando o desempenho alcançado pelas decisões automáticas não é satisfatório, decide-se sacrificar a

transparência em favor do desempenho. O usuário, neste caso, pode estudar o comportamento da aplicação e dirigir a adaptação.

Logo, as políticas podem ser de natureza **global**, válida para toda aplicação; ou **específicas**, válidas para os componentes em particular. Exemplos de política global são:

- *DISCONNECTION migrate nextServer* - indica que antes da operação de desconexão ser realizada as atividades no nodo devem ser migradas para o servidor mais próximo;
- *DISCONNECTION buffering* - indica para chavear as operações de E/S para *buffer* local antes de desconectar.

Políticas específicas são semelhantes às globais, somente identificando o componente (ente) ao qual se referem. Por exemplo,

- *CLONING ente1 anchorOn being:ente2* - indica que a criação do *ente1* deve ser realizada no nodo onde o *ente2* está localizado, e se este migrar, o *ente1* migra junto;
- *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 (*static*). Entes que são declarados como estáticos normalmente estão associados a recursos locais que estes gerenciam;
- *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 de descoberta de recurso do `ISAMcontextService` para encontrar o nodo com o arquivo que preenche os atributos da descrição.

Políticas, além de orientar o EXEHDA indicando **opções de localidade** como nos exemplos acima, também são usadas no sentido de resolver problemas advindos do dinamismo do ambiente, no qual não cabe o conceito de tratamento de exceção – comum em linguagens tradicionais, mas o conceito de **tolerância a falhas**. Por exemplo, no ISAMadapt políticas poderão ser usadas para resolver problemas de sincronização. Considere a situação onde um componente da aplicação pode migrar para outro, mas neste ínterim, este outro componente pode decidir migrar. Para evitar esta situação, usa-se a política específica:

MIGRATION ente1 after ente3

ou a política geral

TIMEOUT <valor>

que estabelece o tempo de espera de sincronização entre componentes, antes de retornar erro para ser tratado pela aplicação (*default* é o bloqueio à espera do componente).

A implementação desta abstração integra o ambiente de programação da linguagem. O menu `Policies` orienta a escolha entre as opções disponíveis. Se nada for especificado, o sistema usa políticas *default*. As decisões informadas pelo programador geram um arquivo *profile* escrito em XML, que será utilizado pelo tradutor ISAMadapt para passar os parâmetros ao ambiente que controla a execução, EXEHDA. Um esboço do menu de políticas globais é apresentado na Figura 4.6. Um exemplo de arquivo XML é dado abaixo.

```
<scheduling>
  <strategy type="TiPS">
    <context v="network::connected">
          useCellScheduler=true
    </context>
    <context v="network::disconnected">
         useCellScheduler=false
    </context>
```

```
</strategy>
<policy>
    <allbeings />
    <max index="CPU_POWER" />
</policy>
<policy composition="append">
  <being type="worker" />
  <range sensor="FREE_PHYS_MEM" lbound="100" ubound="infinity" />
</policy>
<index name="CPU_POWER">
   <switch>
     <sensor name="HOST_BENCH[bogomips]" scale="10" />
     <sensor name="HOST_BENCH[linpack]" scale="4.75" />
     <composite type="sum">
       <sensor name="FREE_PHYS_MEM" />
       <sensor name="HOST_BENCH[scimark]" scale="4.75" />
     </composite>
   </switch>
 </index>
</scheduling>
```

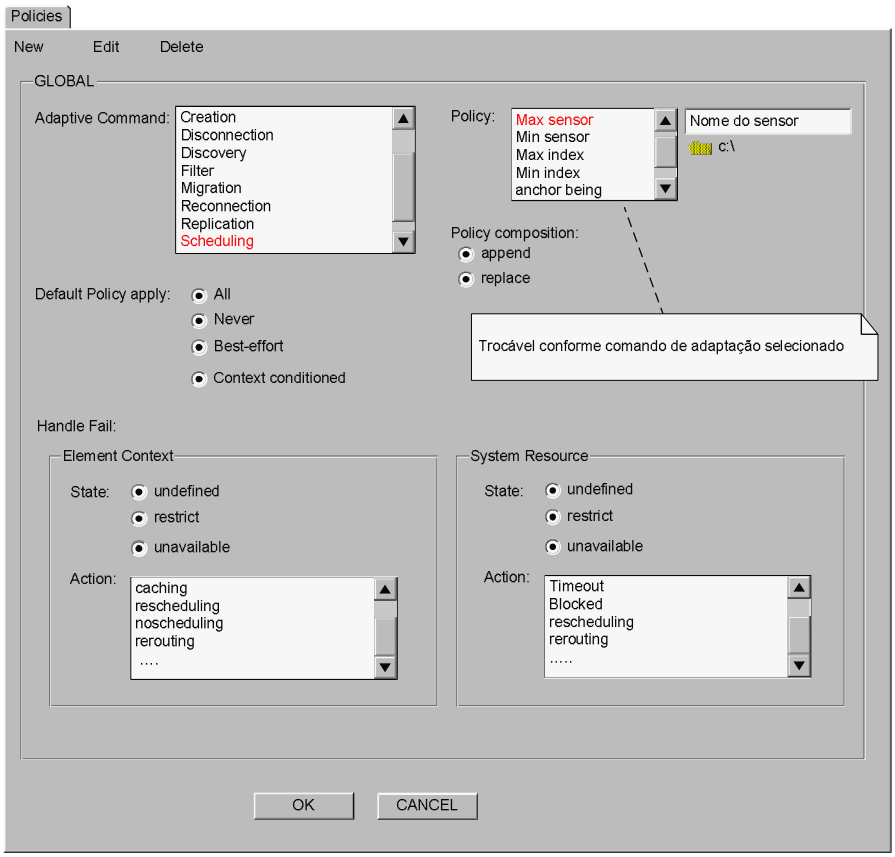


Figura 4.6: Esboço do Menu `Policies` do `ISAMadapt IDE`

4.3.2 Fase de Execução

Cada abstração expressa no código da aplicação tem seu correspondente no ambiente de execução. O contexto relaciona-se com o Serviço de Reconhecimento de Contexto – `ISAMcontextService` (ver Apêndice B); o adaptador relaciona-se com a `ISAMadaptEngine` que gerencia o comportamento adaptativo (ver Capítulo 5), e as políticas relacionam-se com os comandos de adaptação, implementados e gerenciados pelo EXEHDA (YAMIN, 2004).

A aplicação ISAMadapt é uma aplicação gerenciada pelo sistema de execução EXEHDA, o qual é responsável por decisões de gerenciamento físico (dispositivos, recursos e comunicações), mantendo-a com a semântica siga-me, considerando o movimento do usuário e a localidade dos recursos que os componentes da aplicação consomem. Desta forma, o EXEHDA fornece um ambiente de execução *pervasivo* para a aplicação. Maiores detalhes sobre o EXEHDA são obtidos na referência (YAMIN, 2004).

5 DETALHES DE IMPLEMENTAÇÃO

A parte de implementação relativa somente ao ISAMadapt relaciona-se com: interface do ambiente de desenvolvimento, `ISAMadaptEngine` e o tradutor, que traduz código ISAMadapt para Java. Outras frentes do ISAM relacionam-se ao protótipo do `ISAMcontextService` e do *middleware* EXEHDA (YAMIN, 2004). Neste trabalho considera-se a existência do serviço de reconhecimento de contexto, conforme projetado e em fase inicial de implementação (Apêndice B). Salienta-se que o **projeto contextS**, aprovado no edital SEPIN-CNPq-FINEP 01/2002 do Programa de Apoio à Pesquisa, Desenvolvimento e Inovação em Tecnologia da Informação (PD&I-TI), irá aprofundar as questões relativas ao Serviço de Reconhecimento de Contexto nos próximos dois anos.

5.1 Os Pacotes da Implementação

Os pacotes da implementação protótipo da arquitetura ISAM, escritos em Java, estão relacionados no Quadro 5.1. A Figura 5.1 mostra o relacionamento entre esses pacotes. O item 5.4 detalha os pacotes relativos à implementação do ISAMadapt.

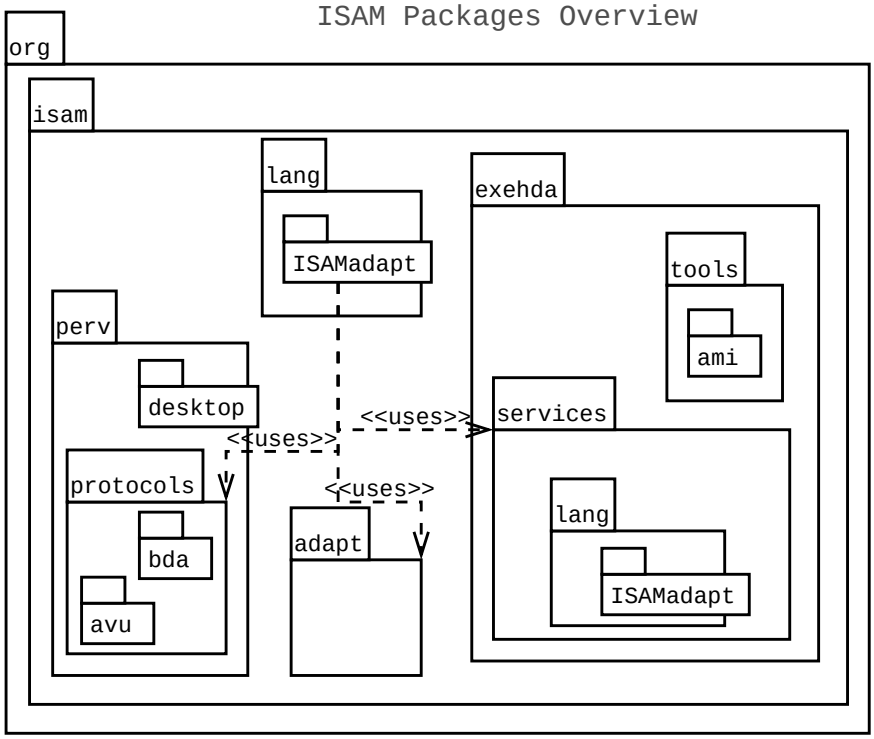


Figura 5.1: Relacionamento entre os Pacotes da Implementação ISAM

org.isam.lang.isamadapt	Define entes, símbolos, tuplas, interface à história dos entes
org.isam.lang.isamadapt.ide	Implementa a ISAMadapt IDE
org.isam.lang.isamadapt.Reflect	Classes que definem reflexão sobre entes e seus métodos
org.isam.lang.isamadapt.Runtime	Classe que define métodos relativos aos comandos da linguagem
org.isam.adapt	Define classes relativas à adaptação
org.isam.adapt.cs	Pacote relativo ao Serviço de Reconhecimento de Contexto na parte de "dados contextualizados"
org.isam.adapt.IsamAdaptEngine	Classe que implementa a máquina de adaptação
org.isam.perv	Define o ambiente pervasivo para o usuário e as aplicações
org.isam.perv.desktop	Define controle sobre o host do usuário
org.isam.perv.protocols	Define protocolos para ambiente virtual do usuário, da aplicação e para acesso ao repositório pervasivo (ISAMbda)
org.isam.exehda	Define as interfaces para todos os serviços do ambiente
org.isam.exehda.ami	Define a interface de gerenciamento da arquitetura
org.isam.exehda.services	Implementa serviços no *core middleware*, tais como serviços de inicialização (sessão, bootstrap,...)
org.isam.exehda.services.cc	Implementa a comunicação e coordenação via Espaço de Objetos
org.isam.exehda.services.lang	Suporte aos serviços específicos de uma linguagem. No caso ISAMadapt: gerência dos entes e de sua história
org.isam.exehda.services.primos	Implementa as primitivas básicas de gerência da mobilidade e da execução
org.isam.exehda.services.tips	Implementam um escalonamento baseado em redes bayseanas

Quadro 5.1: *Toolkit* ISAM

5.2 Programando a Aplicação

A aplicação consciente do contexto é desenvolvida utilizando-se a ferramenta de ambiente de desenvolvimento `ISAMadapt IDE`. Esta ferramenta é uma interface que permite definir o contexto, as alternativas de adaptação, as políticas de adaptação, empacotar a aplicação e enviá-la à `ISAMbda`, e fazer consultas aos componentes nativos fornecidos pela linguagem relativos ao Serviço de Reconhecimento de Contexto. Está prevista a integração do ambiente de desenvolvimento ISAMadapt com o HoloEnv (BARBOSA, 2002). A implementação corrente está desenvolvendo a interface separadamente do HoloEnv, por questões de simplificação, uma vez que o HoloEnv está em constante atualização/evolução. Futuramente, elas serão integradas.

5.2.1 Metodologia Simples

Para entender a dificuldade em construir aplicações *pervasivas* é necessário investigar o processo de projeto de tais aplicações. Como não existem ainda aplicações disponíveis com o perfil das aplicações ISAMadapt, parte-se das conclusões da tese de Dey (2000) relativas às aplicações conscientes do contexto (*context-aware*). Segundo ele, estas são difíceis de construir devido à falta de suporte no nível de infra-estrutura para o projeto dessas aplicações.

O processo do projeto de aplicações *context-aware* sugerido por Dey é composto por:

1. Especificação – especificar, em alto nível, o comportamento sensível ao contexto a implementar, e determinar a coleção de contexto que é requerido para este comportamento ser executável;
2. Aquisição – determinar o hardware/sensores necessário para fornecer o contexto. Para isto, instalar os sensores nas plataformas requeridas, identificar o tipo de dado fornecido pelo sensor, escrever o software que fala com o protocolo usado pelo sensor (se não existir), determinar como consultar o sensor e ser notificado quando ocorre uma troca, armazenar o contexto, interpretar o contexto (se aplicável);
3. Entrega – fornecer métodos para entrega do contexto a uma ou mais aplicações;
4. Recepção – adquirir e trabalhar com o contexto. Para isto, determinar quais sensores são relevantes e como se comunicar com eles, requisitar e receber o contexto, convertê-lo em uma forma usável, analisar a informação para determinar sua utilidade;
5. Ação – executar o comportamento consciente do contexto, se este for útil. Para tal, analisar o contexto, tratando-o como uma variável independente ou combinando-o com outras informações coletadas no presente ou passado, e escolher o comportamento a executar.

Essas etapas foram embutidas nas funcionalidades do `ISAMcontextService`, de forma a simplificar o processo de projeto de aplicações. O programador deve realizar a etapa 1 (especificação), na qual define o comportamento adaptativo da aplicação em alto nível. Para isto, define quem (componentes/entes), com quem (interação), com o que (recursos e contexto) e como (adaptação). As questões quando e onde são determinadas automaticamente pela semântica da execução. Com a ajuda do ambiente de desenvolvimento ISAMadapt, o programador identifica os elementos de contextos disponíveis e os parametriza de acordo com suas necessidades (etapa 2). Se não há um elemento de contexto que atenda às necessidades da aplicação, esta deverá criá-lo e registrá-lo no Servidor. As demais etapas estão embutidas na semântica da execução da aplicação ISAMadapt.

A ação adaptativa é executada automaticamente com base no código de adaptadores, escritos pelo programador, para cada possibilidade de estado do elemento de contexto. Nesta fase do projeto, o programador identifica no código original da aplicação os entes e os métodos que terão comportamento sensível ao contexto, através do atributo *adaptive*, e codifica os adaptadores correspondentes. No código da aplicação também são inseridos os comandos com semântica adaptativa: *move, clone, discovery, reschedule, disconnect, reconnect, install*. Após a codificação, o programador usa o menu `adapters` do `ISAMadapt IDE` para associar o código do adaptador com o estado do elemento de contexto. Para orientar a execução dos mecanismos de adaptação, o programador deverá definir políticas, se não desejar o uso de políticas *default*, escolhidas pelo ambiente de execução EXEHDA. Para tal, usa o menu `policies` do

`ISAMadapt IDE`. Um esboço das estapas que refletem uma metodologia (embrionária) de desenvolvimento de aplicações pervasivas é apresentado na Figura 5.2.

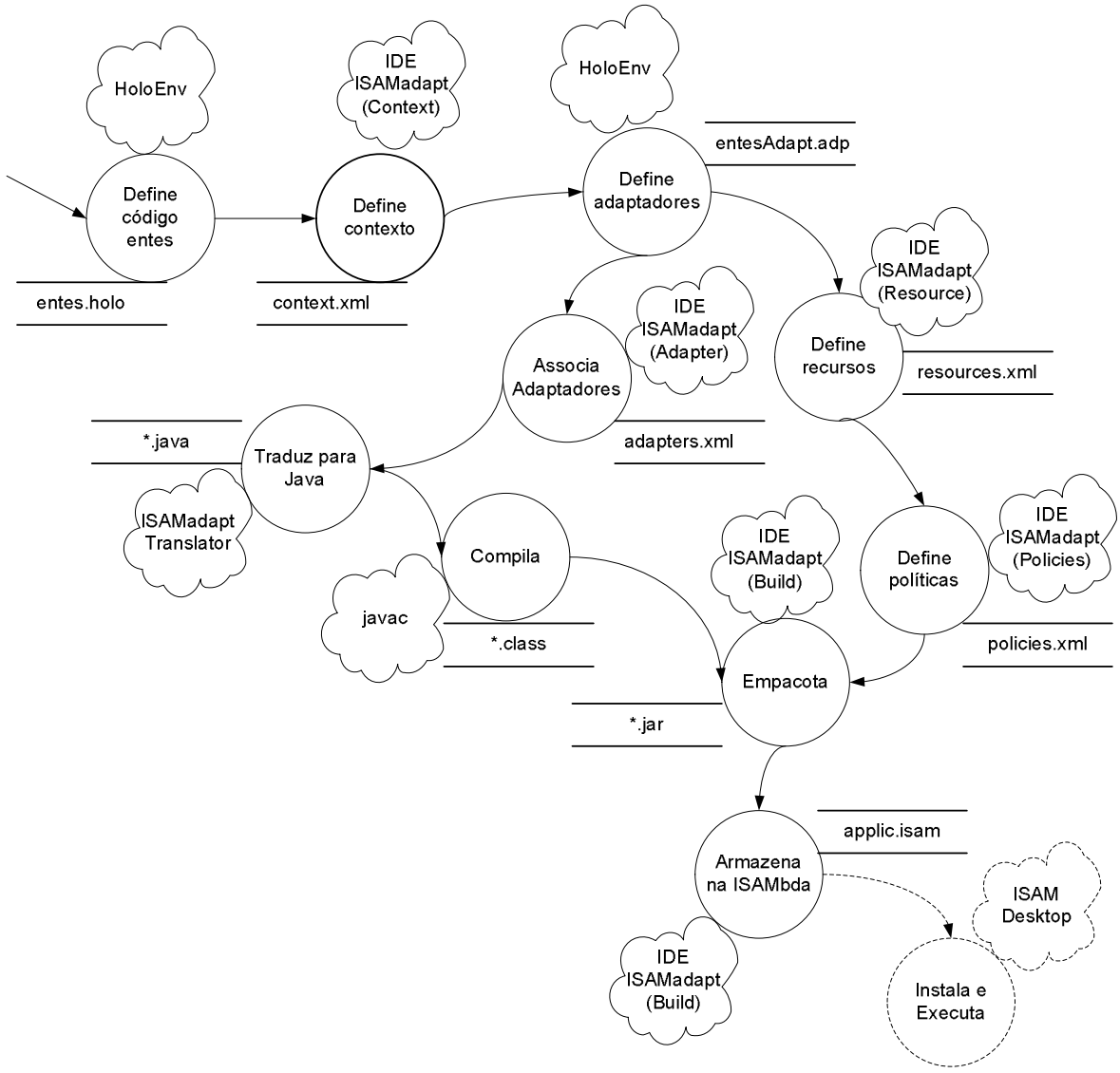


Figura 5.2: Esboço da Metodologia de Desenvolvimento ISAMadapt

Como acontece, em geral, com novos ambientes computacionais, as aplicações pré-existentes (legadas) deverão ser re-projetadas para este novo ambiente. Possivelmente, grande parte das aplicações terá seu código associado a um estado do elemento de contexto e ampliada sua funcionalidade para a adaptação a outros estados dos elementos de contextos em que poderão estar inseridas.

5.2.2 O Ambiente de Desenvolvimento ISAMadapt

A Figura 5.3 mostra a visão dos menus da Interface do Ambiente de Desenvolvimento ISAMadapt. Cada abstração – contexto, adaptadores e políticas – têm um menu de definição na interface, como descrito no Capítulo 4. A interface também permite a consulta às bases de dados do ISAM (`ISAMbda, ava, avu`) referente a essas abstrações.

Como já dito, as informações fornecidas através da interface são armazenadas em arquivos fontes ou arquivos de configuração, escritos no formato XML, em diretórios

que compõem o projeto da aplicação. A interface gera automaticamente os arquivos de configuração, a serem utilizados pelo EXEHDA e pelo tradutor, a partir das informações do programador, usando uma gramática XML (E*xtensible Markup Language*) - formato escolhido por ser largamente usado para troca de dados. Os arquivos gerados são relativos ao contexto, aos adaptadores e às políticas.

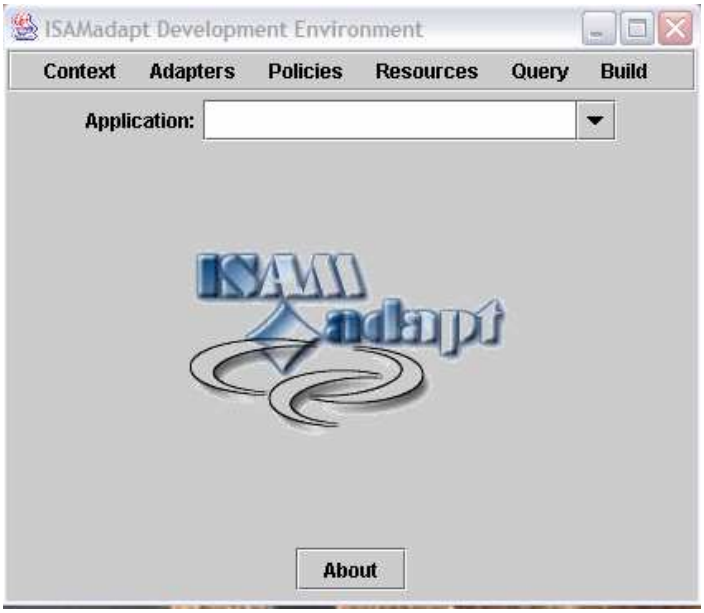


Figura 5.3: Esboço da Interface do `ISAMadapt IDE`

5.2.3 Alterações na Hololinguagem

Procurou-se interferir minimamente na linguagem original ao se definir as construções que seriam inseridas na HoloLinguagem. A aplicação Holo foi alterada para introduzir as abstrações ISAMadapt conforme descrito no Capítulo 4.

Para o desenvolvimento de aplicações, sentiu-se a necessidade de introduzir outras alterações na HoloLinguagem propriamente dita:

a) qualificador *being* para designar os entes. Isto permite diferenciar no código os entes das definições de procedimentos. Na linguagem original, a diferença estava na posição no código, dificultando uma leitura mais direta;
b) entes de sistema (***system beings***) – são entes que implementam a funcionalidade do sistema integrando-as ao ambiente definido, tais como *file being* e *printer being*. Por exemplo, *clone (file, file_id).* O ente *file* implementa a disponibilização de registros em tuplas, que comporão a história do ente *file*, e disponibiliza métodos de acesso e de gravação;
c) história distribuída com *caches* locais, o que permite diminuir a repercussão da desconexão no comportamento da aplicação. A história local é enviada ao servidor na forma "conta-gotas", ou quando da execução do protocolo de desconexão, e enchimento da *cache*. O qualificador *private* foi introduzido para qualificar a história privada do ente que é armazenada localmente. Este indica que a história é somente visível ao próprio ente, e não visível aos seus descendentes. Sintaxe: *private ! <tupla>;*
d) comando *kill* para encerramento forçado de entes. Sintaxe: *kill(ente)*;
e) comando *exit* para encerramento compulsório da aplicação. Sintaxe: *exit*;
f) execução de código nativo Java, principalmente para o tratamento de interfaces gráficas. Sintaxe: *native Java {* <blocos_comandos>* }.*

A gramática da HoloLinguagem original (BARBOSA, 2002) foi modificada para introduzir as regras de declaração de contexto, procedimentos adaptativos, alterações nos comandos para contextualizá-los, e inserção de novos comandos que expressam adaptação. A Figura 5.4 ilustra as regras inseridas ou modificadas, escritas em BNF estendida utilizada no javaCC.

5.2.4 Compilando a Aplicação

A estratégia usada pelos autores para disponibilizar rapidamente a linguagem é traduzi-la para a linguagem Java. Para isso, usa-se a gramática Holo + ISAMadapt como entrada para a ferramenta JavaCC, e obtém-se um tradutor ISAMadapt. Este é usado para receber como entrada o programa Holo+ISAMadapt e os adaptadores, gerando o programa Java correspondente. O tradutor insere no código Java as chamadas necessárias ao `ISAMcontextService`, `ISAMadaptEngine` e ao suporte EXEHDA à linguagem. Esta solução permite-nos utilizar recursos de Java como *threads*, e carga dinâmica de código na implementação do ISAMadapt.

Java é desenvolvida sob o espírito da mobilidade: "*write once, run anywhere*". Além disso, a escolha de Java se deve a três outros motivos:

a) traduzir Holo para Java foi a estratégia usada pelo autor da Hololinguagem para uma prototipação rápida da linguagem (BARBOSA, 2002);
b) Java exibe muitas características que facilitam o desenvolvimento de aplicações móveis. Entre elas, a portabilidade dos *bytecodes* permite a execução em diferentes hosts com diferentes arquiteturas de software e hardware; *multithreading* torna os agentes colaborativos e autônomos; serialização de objetos suporta a migração e persistência; carga de classe dinâmica permite o projeto de estratégias eficientes para a recuperação de código dos entes;
c) a versão *Micro Edition* de Java para uso em PDA's mantém as principais características Java (ver Apêndice A).

A Figura 5.5 relaciona os componentes envolvidos no processo de compilação e indica como eles são gerados.

Durante a programação, o programador usando o ambiente de programação ISAMadapt gera código Holo + ISAMadapt; gera código dos adaptadores, pesquisa as opções disponíveis no ambiente; define os elementos de contexto e as políticas de adaptação. O ambiente de desenvolvimento ISAMadapt gera os arquivos *.holo, *.adp, *.xml correspondentes.

Na compilação, os arquivos XML relativos ao contexto são varridos (*parsing*) para comporem as informações necessárias à tradução. A compilação da aplicação é realizada em duas etapas: uso do pré-processador/tradutor e uso do compilador Java padrão. O processo de tradução – que traduz código Holo + ISAMadapt em código Java – é a atividade que junta as várias informações fornecidas pelo programador em uma unidade integrada. Do processo de compilação resultam: código Java com a funcionalidade da aplicação e chamadas ao middleware; código Java dos adaptadores com chamadas ao middleware; tabelas de configuração do ambiente de execução.

Gramática ISAMadapt na notação BNF estendida usada pelo JavaCC:
(...)? - opcional ocorrência de (...)* - zero ou mais ocorrências de (...)+ - 1 ou mais ocorrências de
(... | ...) - escolha entre [...] - combinação entre ~[...] - combina exceto

```
<PROGRAM>: ( <HOLO> | <ADAPTER> )
<HOLO>:  "holo" "{" <SEQ> "}" (<BEINGS>)* "}"
<BEINGS>: ( "being" <BEING_NAME> "{" <CONSTRUCTOR> <BLOCK>
                        (<PROCEDURE_DECL>)* (<HISTORY>)* "}"
                | (<CONTEXT_DESCR>)?  "adaptive" "being" <BEING_NAME>  ("param" <LIST_PARAM>;)?
                  "context" <CONTEXT_NAME> ( "::" <CONTEXT_STATE_LIST>)?
                  "{" (<ADAPTIVEMETHOD_DECL>)* (<PROCEDURE_DECL)* (<HISTORY>)* "}")
<ADAPTER>: (<BEING_ADAPTER> | <METHOD_ADAPTER> )
<BEING_ADAPTER>: <CONTEXT_DESCR>)? "adapter" "being" <BEING_NAME>
          ("::"<CONTEXT_STATE_NAME>)?   "{" <SEQ> "}"
<METHOD_ADAPTER>:  <CONTEXT_DESCR>)?  "adapter"  <BEING_NAME> "."
        <METHOD_NAME>  ("(" <VARIABLE_LIST> ")")? ("::"<CONTEXT_STATE_NAME>)?  <BLOCK>
<CONTEXT_DESCR>: ( "//@context:" <CONTEXT_NAME> ( "::" <CONTEXT_STATE>)? <EOL> |
            "//@description:" <STRING> <EOL> |   "//@file:" <STRING> <EOL> )*
<CONTEXT_STATE_LIST>: "(" <CONTEXT_STATE_NAME>   ("," <CONTEXT_STATE_NAME>)* ")"
<SEQ>: (<ADAPTIVEMETHOD_DECL>)* <CONSTRUCTOR> <BLOCK>
        (<PROCEDURE_DECL>)* (<HISTORY>)*
<CONSTRUCTOR>: <CONSTRUCTOR_NAME> "(" (<VARIABLE_LIST>)? ")"
<BLOCK>: "{" (<COMMAND> ";" )* "}"
<ADAPTIVEMETHOD_DECL>: "adaptive" <PROC_NAME>   "(" ( <PARAMETER_DECL_LIST> )? ")"
                                "context" <CONTEXT_NAME> ("::" <CONTEXT_STATE_LIST>)? ";"
<PROCEDURE_DECL>: <PROC_NAME> "(" (<PARAMETER_DECL_LIST>)* ")"  <BLOCK>
<COMMAND>:  (  <NATIVE>       | <CONTEXT>     | <ONCONTEXT>          | <MOVE>
              | <CLONE>       | <CALL_PROCEDURE>                     | <IF>
              | <WHILE>       | <IN>          | <OUT>                | <READ>
              | <BLOCK>       | “exit”        | <KILL>               | <PREFETCH>
              | <INSTALL>     | "disconnect"  | "reconnect"          | <RESCHEDULE>
              | <DISCOVERY>   | “push”  FILE_LIST to  (<idUSER> |  “device” )          |  ... )
<NATIVE>: "native" "Java" "{*" <JAVA_CODE> "*}"
<CONTEXT>: "context" <CONTEXT_NAME> ("::" <CONTEXT_STATE_LIST> )?
<ONCONTEXT>: ("sync")? "onContext" <CONTEXT_NAME> ( "::" <CONTEXT_STATE_LIST>)?  <BLOCK>
 <MOVE>: "move" ("to" | "closeTo" ) <DESTINY>
<DESTINY>:: ( "being:" <STRING> | "parent"  |  “env:”  <STRING> |
              "resource:”  <STRING>  | <VARIABLE> ) )
<CLONE>: "clone" "(" <CLONED> "," <NEW_BEING_ID> ")"  ( ("on" | "anchorOn" | "closeTo" ) <DESTINY> )?
<CLONED>:  <BEING_NAME> ( "(" <PARAM_LIST> ")")?
<IN>: <HIST> ("#" | "#?") <GABARIT>    // retirada bloqueante ou não-bloqueante
<READ>: <HIST> ("." | ".?") "(" <GABARIT>  ")"  // leitura     bloqueante ou não-bloqueante
<GABARIT>:  ( <STRING> | <VARIABLE>  | "%" <VARIABLE> )
            ( "," (<STRING> | <VARIABLE> | "%" <VARIABLE> ) )*
<OUT>: <HIST> "!" <TUPLE_HISTORY>   // escrita na história
<HIST>: ( "history" | "private" | "parent" | "any")
<TUPLE_HISTORY>: <VARIABLE> | STRING | "tuple (" <TUPLE_HISTORY> ( "," <TUPLE_HISTORY> )* ")"
<KILL>: "kill" "(" (BEING_NAME> )? ")"
<PREFETCH>: “prefetch” <FILE_LIST>  “to”  (“device” |  <idUSER> )
<INSTALL>: "install" "(" <DIRECTORY> <APLIC_NAME> ")" at  <IDUSER>  ("[start]")?
<RESCHEDULE>: "reschedule" ( <WHO>)? ( "(policy:" <RULE_NAME> ")" )?
<WHO>: ( "all" | "begin:"<BEING_NAME> )
<DISCOVERY>: "discovery"  <VARIABLE> "(" <PROTOCOL_DEF> ")"
          ( "," "discovery" <VARIABLE> "(" <PROTOCOL_DEF> ")" )*  <BLOCK>
<PROTOCOL_DEF>: ( "avu:" | "ava:" | "resource:" | "service:" ) "//" <STRING> ".rsc"
<CALL_PROCEDURE>: <STRING> "(" <PARAMETER_LIST> ")"
<WHILE>: "while" "(" <COND> ")" <BLOCK>
<COND>: <CONDITION> ( <OPER> <CONDITION> )*
<CONDITION>: ( "onContext" <CONTEXT_NAME> ( "::" <CONTEXT_STATE_LIST> )?    | .... )
<FILE_LIST>: <STRING> ( "/" <STRING> )* ( "," <STRING> ( "/" <STRING> )* )*
<HOSTNAME>: ( <STRING> ("." <STRING> )* | "myHost" | "myHome" | ...)
<ATRIB_LIST>: ( <VARIABLE> "=" <VALUE> ( "," <VARIABLE> "=" <VALUE> )*
<HISTORY>:  ( "history" |  "(private)" ) "{" ( <CLEI> )* "}"
```

Figura 5.4: Gramática Básica ISAMadapt

5.2.5 Preparando a Aplicação para Execução

O menu `Build` do `ISAMadapt IDE` (Figura 5.6) auxilia no empacotamento da aplicação, na geração do descritor da aplicação *.isam (ver exemplo no Capítulo 6, Figura 6.6), e no envio da aplicação empacotada para o repositório de aplicações (`ISAMbda`). Para manter consistência, esta opção é somente disponível para os usuários cadastrados para tal.

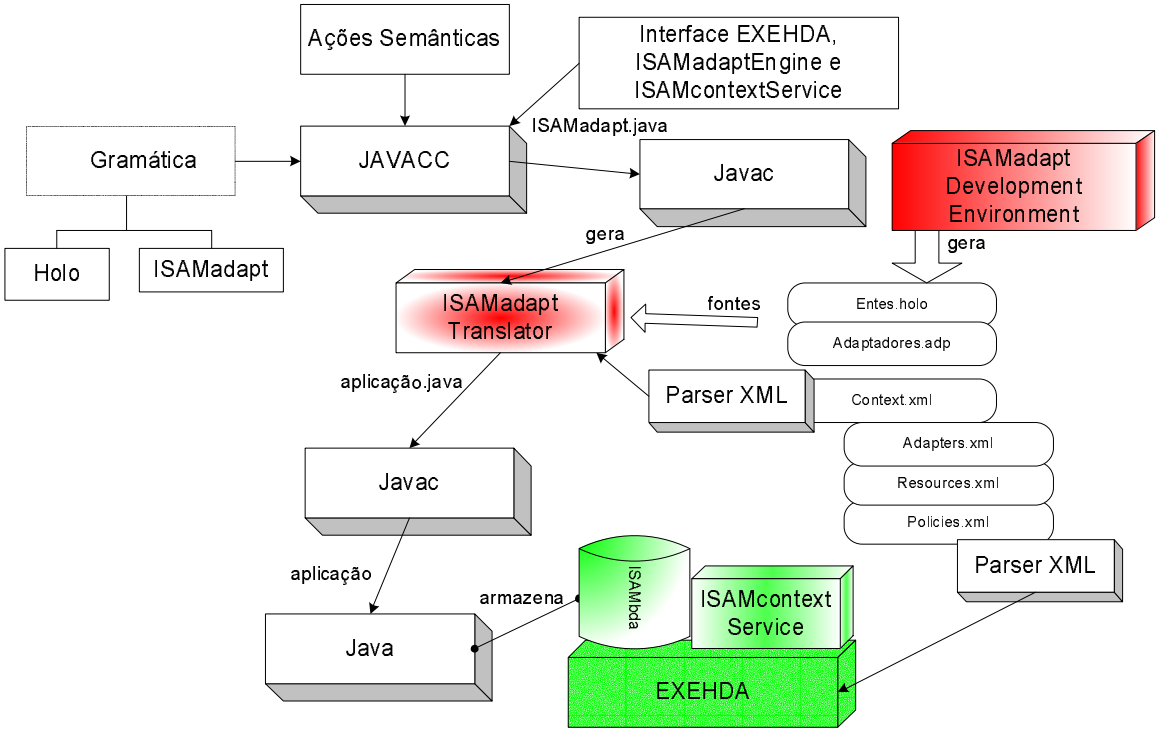


Figura 5.5: Componentes do Processo de Compilação

5.3 Executando a Aplicação

A aplicação ISAMadapt executa sob o controle do ambiente de execução EXEHDA, o qual simula um sistema operacional *pervasivo*. Este é responsável por gerenciar o Ambiente Virtual do Usuário e da Aplicação, seus dados, recursos e códigos armazenados na `ISAMbda`. Quando o sistema inicia (*start*) uma aplicação, este carrega as políticas de adaptação correspondentes, configura o `ISAMcontextService` e dispara a aplicação. O *runtime* da linguagem (`ISAMadaptEngine`) gerencia a adaptação no nível da aplicação, comunicando-se diretamente com o `ISAMcontextService`. O sistema EXEHDA entra em ação novamente quando um ente é criado ou migrado, ou quando outro mecanismo de adaptação é utilizado, caso em que as políticas definidas pela aplicação são avaliadas.

Se por questões de gerenciamento (balanceamento de carga ou deslocamento dos recursos para o centro geométrico mais próximo ao usuário), o EXEHDA decide migrar fisicamente alguns entes da aplicação, a lógica (comportamento) da aplicação não é alterada. Somente um rearranjo nos serviços de localização e *naming*, internos ao EXEHDA, é realizado de forma transparente para a aplicação. Aplicações são construídas considerando a camada *middleware* fornecida pelo EXEHDA, que implementa os serviços básicos da arquitetura ISAM. A adaptação que é realizada nesta camada reflete indiretamente na aplicação.

Outra estratégia de adaptação presente no ambiente de execução é obter o **código sob demanda**. Esta é uma forma natural de manter os entes com tamanho reduzido e reduzir o tráfego na rede. Outras vantagens para o ambiente móvel são: (a) reduz o impacto dos recursos limitados do dispositivo móvel, desde que o software não necessita ficar armazenado localmente; (b) evita a instalação de novas versões, já que esta será feita automaticamente, para o ambiente caracterizado pela alta escalabilidade (número elevado de clientes móveis); (c) permite minimizar a complexidade e tamanho do sistema e reduzir a dependência de recursos limitados dos computadores móveis.

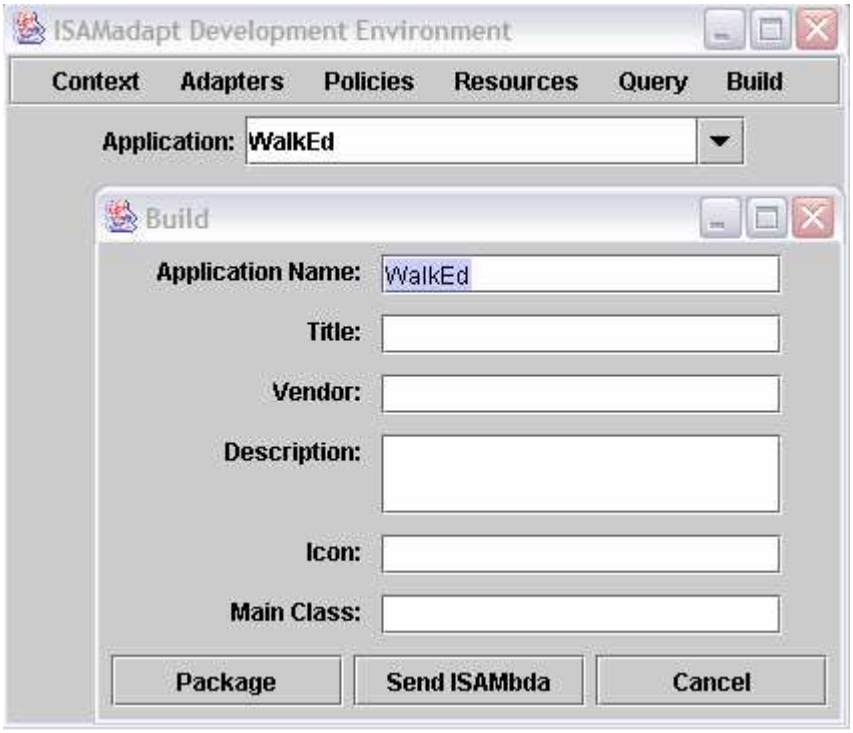


Figura 5.6: `Build Menu` do `ISAMadapt IDE`

5.4 Detalhes da Implementação da ISAMadaptEngine

A `ISAMadaptEngine` é responsável por gerenciar a adaptação ao contexto dos entes e métodos da aplicação, e pelos comandos contextualizados. Suas principais funções são: receber a notificação de alteração no estado do elemento do contexto, selecionar entre as alternativas e reconfigurar o código da aplicação. Em cada nodo do ambiente *pervasivo*, ISAMPE, que contém entes em execução será criada uma instância da `ISAMadaptEngine` para gerenciá-los.

A máquina de execução ISAMadapt opera em conjunção com o Serviço de Reconhecimento de Contexto numa relação baseada no modelo *publish-subscribe*: a `ISAMadaptEngine` registra o interesse dos entes da aplicação, solicita reconfiguração do Serviço conforme necessidades da aplicação, recebe a notificação do ISAMcontextServer, ativa/desativa o recebimento de notificação.

A reconfiguração do Serviço relaciona-se com a publicação nas tabelas do Registro (componente `register` do `ISAMcontextService`) de novos códigos de sensoramento, agregação e interpretação que atendam melhor aos requisitos da aplicação, ou com a modificação de parâmetros relativos aos elementos de contexto previamente definidos. A carga do código destes componentes, que darão a visão

particular do contexto para a aplicação, é dinâmica, e realizada quando da ativação destes pelo componente `register` do `ISAMcontextService`. Todos os códigos fontes da arquitetura ISAM estão armazenados na base de dados *pervasiva*, `ISAMbda`, gerenciada pelo componente `launcher` do sistema de execução EXEHDA.

A adaptação da aplicação consiste na carga sob demanda do código dos adaptadores, correspondente ao estado corrente do elemento de contexto em foco. Este procedimento é executado a cada notificação de alteração no contexto. Existe um objeto `ISAMadaptEngine` por aplicação no nodo. Este mantém uma lista dos entes inscritos para serem notificados da alteração de contexto (executaram o método `addContextListener`). O objeto `ISAMadaptEngine` recebe os eventos de alteração de contexto. Do ponto de vista dos adaptadores, ela é a fonte dos eventos de modificação de contexto. Como ocorre isso:

* na criação (instanciação), o objeto adaptador se registra junto à `ISAMadaptEngine` local indicando assim sua intenção em receber determinados tipos de eventos de contexto (`addContextListener`);

* na chegada de um evento gerado pelo notificador, a `ISAMadaptEngine` local faz uma difusão (*multicast*) para todos os adaptadores registrados, invocando o método `contextChanged` daqueles adaptadores;

* em decorrência disso, o adaptador altera zero ou mais instâncias de objetos-implementação;

* chamadas de método normalmente não passam pela `ISAMadaptEngine`, o que seria mais eficiente.

A implementação da abstração e controle da adaptação usa os padrões de projeto: estruturais: *adapter* e *bridge*; comportamentais: *strategy* e *observer* (GAMMA; JOHNSON; HELM; ULISSIDES, 2000). Além disso, usa-se herança múltipla de interfaces, e delegação para modelar o comportamento de adaptação dinâmica expresso na linguagem. A Figura 5.7 esboça a solução da adaptação em UML, aplicando os padrões acima. Esta solução é explicada na Seção 5.3.2.

Os principais métodos definidos na interface da `ISAMadaptEngine` (Figura 5.8) são:

- `getBeingImpl` – retorna um objeto da classe que implementa o ente correspondente ao contexto/estado notificado à aplicação;
- `getMethImpl` – retorna um objeto da classe que implementa o método correspondente ao contexto/estado notificado à aplicação;
- `addContextListener` – configura o Serviço de Contexto, conforme descrito em `context.xml`, inscreve-se para receber notificação de alteração de contexto, e adiciona o ente à lista de inscritos. Esta operação é chamada por demanda da aplicação, na criação do ente, ou pela execução da operação de migração do ente de outra célula;
- `removeContextListener` – remove a inscrição de notificação do Serviço e retira o ente da lista de inscritos. Esta operação é chamada quando da migração do ente para outra célula, ou seu término;
- `enableNotifications` – habilita/desabilita o recebimento de notificação de troca de contexto pela aplicação. A `Engine` recebe a notificação, mas não a propaga (logo, não há adaptação do ente). Razões para não desabilitar no Serviço: coletor é para várias aplicações; contexto sempre requer o atualizado; tempo de ativação é alto; notificação pode ser retornada a qualquer momento requerendo contexto atualizado; a aplicação pode continuar a fazer consulta sob demanda;

- `checkContext` – verifica o estado do contexto corrente forçando uma avaliação;
- `getContext` – obtém o estado corrente do contexto, ultima informação mantida no Servidor de Reconhecimento de contexto;
- `waitForContext` – bloqueia o ente à espera da informação de estado do contexto. Usada na implementação da semântica do comando `onContext` sincronizado;
- `runOnContext` – registra uma thread (*action*) para receber notificação de contexto, suspende a execução até que o estado do contexto desejado seja observado. É usado na implementação da semântica do comando onContext assíncrono.

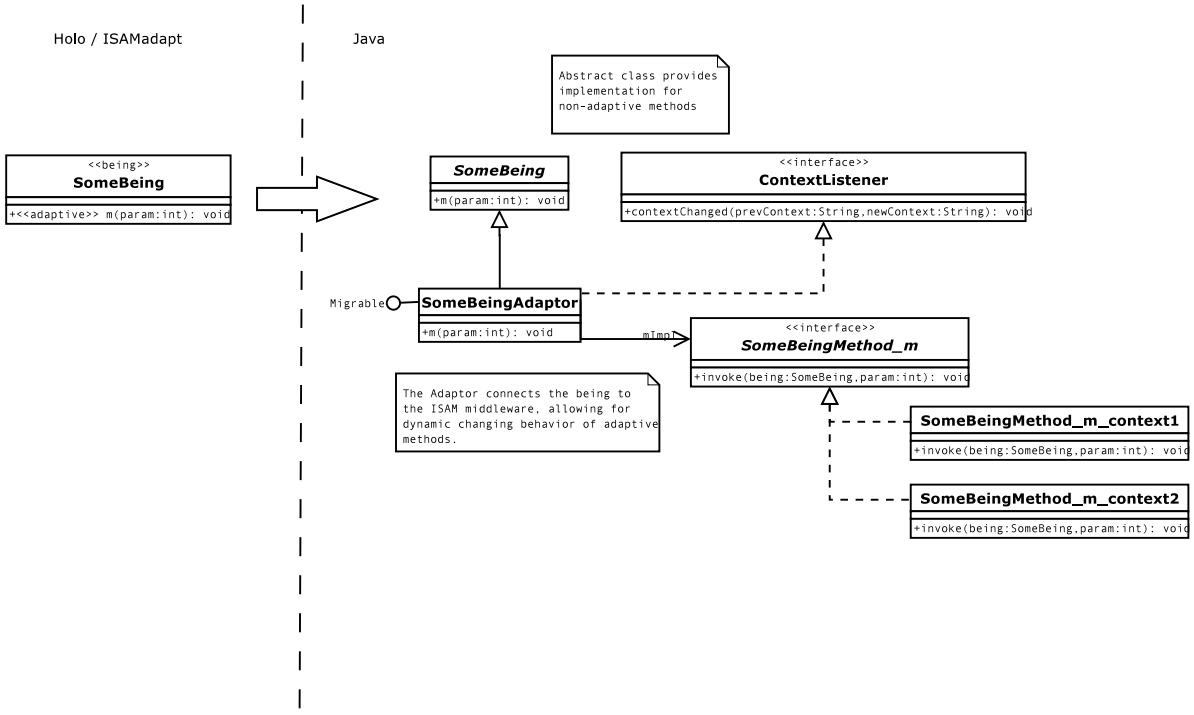


Figura 5.7: UML da Adaptação

```
package org.isam.adapt;
public class IsamAdaptEngine
{
Object getBeingImpl(String beingClass);
Object getMethImpl(String beingClass, String methodName);
void addContextListener(ContextListener l, Context ctx);
void removeContextListener(ContextListener l);
Context checkContext(Context ctx);
Context getContext(Context ctx);
void waitForContext(Context ctx);
void runOnContext(Context ctx, Runnable action);
void enableNotifications (boolean enable);
ResorceDesc[] discoverResource(String description);
}
```

Figura 5.8: Classe Java `ISAMadaptEngine`

5.5 Detalhes da Implementação do ISAMadaptTranslator

Esta Seção descreve o mecanismo de tradução das construções ISAMadapt para código Java, o qual é baseado nas classes que implementam a máquina de adaptação (Figura 5.8) e o suporte em tempo de execução à linguagem (Figura 5.9).

5.5.1 Regras de Tradução para a Geração de Código Java

A implementação da adaptação é explicada a seguir, onde o mecanismo de geração de código do ISAMadapt é detalhado para os casos: ente adaptativo, e método adaptativo. Na seqüência são detalhadas as regras de tradução para os demais comandos ISAMadapt.

5.5.1.1 Entes Adaptativos

Durante a tradução para a linguagem Java, cada ente adaptativo dá origem a:

- uma interface, que define os métodos do ente;
- uma classe `BeingNameAdapter` que implementa essa interface, e também as interfaces `org.isam.adapt.ContextListener` e `org.isam.exehda.primos.Migrable`;
- uma classe `BeingNameImplBase` que provê implementações para os métodos não-adaptativos definidos no ente (desta forma estes podem ser reescritos ou tornados também adaptativos).

```
package org.isam.lang.isamadapt;
public class Runtime
{
  static void disconnect();
  static void reconnect();
  static void install(String applicName, Symbol idUser);
  static void prefetch(Object[] file);
  static void push(String scrResourceURL, String targetResourceURL);
  static void exit();
  static void killBeing(String beingName);
  static Symbol createBeing(ClassBeing being, Being id,
                            Object [] param, Object hints);
  static void moveBeing(ClassBeing dest);
  static void reschedule(Object hints);
  static Symbol getEnv(String varName);
  static History getHistory();
  static History getChildHistory(Symbol childName);
  static History getParentHistory(String child);
}
public  interface History
{  Tuple in (Tuple pattern, Long waitmilis);
   Tuple in_r ((Tuple pattern, Long waitmilis);
   Tuple out (Tuple t);
}
```

Figura 5.9: A Classe Java `isamadapt.Runtime`

Por exemplo, considere o trecho de código ISAMadapt na Figura 5.10 cujo código Java correspondente é expresso na Figura 5.11. Observe que as chamadas de método são redirecionadas para o objeto que efetivamente provê uma implementação do ente para um dado contexto de execução, o qual encontra-se armazenado no atributo `beingImpl` do ente `BeingNameAdapter` (delegação).

Note que chamadas a métodos adaptativos definidos na interface do ente são delegadas ao ente `BeingNameAdapter`. O parâmetro `adapter` é passado pela `ISAMadaptEngine` quando da criação do ente `BeingNameAdapter` para permitir o acesso ao ente associado para chamada a métodos definidos neste.

Métodos não adaptativos são chamados diretamente. Esta organização faculta ainda a definição de novos métodos adaptativos em uma dada implementação alternativa do ente (`adapter`), além dos originalmente definidos no ente-base.

O ente-adaptador gerado, por implementar a interface `ContextListener`, torna-se apto a receber os eventos de modificação de contexto originados pela `ISAMadaptEngine` e, por conseguinte, proceder às respectivas adaptações de código.

5.5.1.2 Métodos Adaptativos

Na geração de código para os métodos de comportamento adaptativo, cada método adaptativo do ente dá origem a uma interface `<tipoEnte>Method_<nomeDoMetodo>`, a qual define um único método *`invoke`* e deve ser implementado por todas as versões definidas para aquele método adaptativo.

Por exemplo, considere o fragmento de código ISAMadapt da Figura 5.10 e o código traduzido para Java da Figura 5.11. Os parâmetros, definidos como sendo do tipo `Symbol`, correspondem aos parâmetros do método adaptativo original. No que se refere ao ente-adaptador gerado, este inclui um atributo do tipo da interface definida para o método, redirecionando sua implementação do método para o objeto referenciado pelo atributo correspondente (delegação).

```
adaptive being BeingName
       context cxt
       param A,B,C
{
   // declaração métodos adaptativos
   adaptive mAdp(param) context ctx2;

   // definição  métodos não-adaptativos
   m1(param)
       { // code ...}

   history {...}
}
```

Figura 5.10: Fragmento de Código do Ente Adaptativo

5.5.1.3 Adapters

A implementação de cada uma das possibilidades de comportamento adaptativo é realizada pelo código do adaptador de métodos (*.adp). Implementações alternativas para o ente adaptativo (`adapter`), quando transpostas para Java, resultam em classes que herdam da implementação base do ente.

No caso do ente `BeingName` dado como exemplo na Figura 5.10, a classe `adapter` correspondente teria a forma da Figura 5.12. Na tradução para Java, o trecho de código da Figura 5.12 é transformado no código Java mostrado na Figura 5.13.

```
interface BeingName
{ void   run();  //constructor
  void mAdp(Symbol param);
  void   m1();
}
interface BeingNameMethod_mAdp
{ void invoke (Symbol param);
}
class BeingNameAdapter
   implements BeingName, Migrable, ContextListener,
{  private BeingName beingImpl;
    BeingName adapter;
    History history;
    private BeingNameMethod_mAdp methImpl;
    Symbol A;
    Symbol B;
    Symbol C;
  // constructor
  BeingNameAdapter (adapter, Symbol A, Symbol B, Symbol C)
    { beingImpl=IsamAdaptEngine.getBeingImpl(“BeingName”);
    IsamAdaptEngine.addContextListener(this, Context cxt);
    this.adapter=adapter;
    history=Runtime.getHistory(); // cria buffer local
    methImpl=IsamAdaptEngine.getMethImpl
             (“BeingName”,” methodName”);
    IsamAdaptEngine.addContextListener
             (this, Context cxt2);
    this.A=A;
    this.B=B;
    this.C=C;
     }
//implem. métodos delega ao beingImpl
 m1() {beingImpl.m1();}
 run() {beingImpl.run();}
 mAdp(Symbol param) { methImpl.invoke (Symbol param);}
}
contextChanged(Context newC)
{ if newC.getType()== ctx
    {beingImpl=ISAMadaptEngine.getBeingImpl
            (“BeingName”);
      }
  if newC.getType()==ctx2
   {methImpl=IsamAdaptEngine.getMethImpl(“BeingName”,” methodName”);
}
captureContext() {return null;}
restoreContext() {return null;}
}
class BeingNameImplBase
    implements BeingName, Externalizable
{ // implementacao dos métodos não adapt
  m1 (Symbol param)
     {...// translated code }
}
```

Figura 5.11: Fragmento de Código Java Gerado para o Ente Adaptativo

```
adapter being BeingName(param)::ctxState
{// constructor
adaptive methA(param2) context ctx3;
 …. code }
  //definição métodos não-adaptativos
   m1();
}
```

Figura 5.12: Fragmento de Código do Adaptador

```
interface BeingNameMethod_methA
{ void invoke (Symbol param2);
}
class BeingNameImplBase_ctxValue
   implements Migrable, ContextListener
   extends BeingNameImplBase
{
   private  BeingName adapter;
   private BeingNameMethod_methA methImpl;
   Symbol param;
   //contructor
   BeingNameImplbase_ctxValue (BeingName adapter, Symbol param)
{  this.adapter=adapter;
          this.param=param;
// acesso métodos compartilhados definidos no ente
//  adapter.methodName(param);
    methImpl=IsamAdaptEngine.getMethImpl
             (“BeingName”, “methName”);
}
//métodos não-adaptativos do adapter
m1() {…… }

methA (Symbol param)
     { methImpl.invoke(Symbol param2);}
readExternal  (Symbol obj) { // save history }
writeExternal (Symbol obj) { // restore history }
}
```

Figura 5.13: Fragmento de Código Java Gerado para o Adaptador

O trecho de código do adaptador de método

```
adapter beingNameMethod.nameMethod(param)::ctxValue
{
      //code
}
```

é traduzido para o código:

```
class BeingNameMethod_nameMethod_ctxvalue
      implements BeingNameMethod_nameMethod
{
   void invoke (Symbol param)
   { ... // code ... }
}
```

ISAMadapt tem, por definição, três elementos para tratar a adaptação: ente, método e adaptador. Logo, pode-se ter as combinações enumeradas no Quadro 5.1.

ENTE	NÃO ADAPTATIVO	Método não-adaptativo
		Método adaptativo
	ADAPTATIVO	Método não-adaptativo
		Método adaptativo
ADAPTER	Método não-adaptativo	
	Método adaptativo	

Quadro 5.1: Combinações de Elementos Adaptativos

Existe a possibilidade de o ente adaptativo fazer chamada a métodos adaptativos desde que seja adaptativo a outro elemento de contexto. Por exemplo, o ente adapta-se ao tipo de dispositivo, e um de seus métodos compartilhados adapta-se à flutuação da energia/bateria.

5.5.1.4 Comando onContext

A tradução do comando `onContext` segue conforme abaixo. Salienta-se que o método `runOnContext`, antes de anexar a *thread* na lista de inscritos à notificação de contexto, verifica se este já não está disponível usando o método `checkContext.` O trecho de código a ser gerado é:

```
// assynchronous wait,
// creates the thread and attach to IsamAdaptEngine
   Context context = new Context (ctx, state);
   Runnable ctxAction= new Runnable {
      void run () {
                        ... code translated to  Java
                  }
   }
   IsamAdaptEngine.runOnContext (context, ctxAction);
```

No caso de ser um comando síncrono (qualificador `sync`), o método `waitForContext` adiciona o ente à lista de inscritos a receber notificação de alteração de contexto, e suspende sua execução:

```
   Context context = new Context (ctx, state);
   if (IsamAdaptEngine.waitForContext(context) == timeout)
         { //envia mensagem ao usuário
            exit();
         }
   ...  code translated to Java
```

5.5.1.5 Outros comandos ISAMadapt

Os comandos de adaptação `move, clone, discovery, disconnect, reconnect, reschedule` são implementados pelo suporte EXEHDA à linguagem. O código ISAMadapt gera uma chamada aos métodos que implementam tais comandos, passando os parâmetros e atualizando os atributos necessários. Por exemplo, a tradução do comando `clone:`

```
  clone (worker, worker_id) [ (on  | anchorOn | closerTo) Location ] ;
```

resulta na chamada:

```
isamadapt.Runtime.createBeing(class typeBeing, String idBeing,
                              Symbol param, Object schedulingHints);
```

A implementação do comando segue a semântica definida no ISAMadapt (Capítulo 4) e é decomposta nas seguintes chamadas:

```
-> Runtime.createBeing(...)
          -> primos.createObject(...)
                -> tips.chooseCreationHost()
        // se não informado schedulingHints
                -> construtor do ente, que chamará
                     ->ISAMadaptEngine.getBeingImpl(...)
                     ->ISAMadaptEngine.addContextListener(...)
               -> ativador do ente
                     -> exehda.cc.createTupleSpace(...)
                     -> exehda.hm.createBeing(...)
               ->ente.start()
```

Os demais comandos seguem essa estratégia e são traduzidos para chamadas do *middleware* de execução, conforme Quadro 5.2.

5.5.1.6 Manipulando a História (History)

O ente ao ser criado obtém um objeto da classe `History`, o qual cria um *buffer* local para armazenar as tuplas geradas pelo ente. Quando o ente migra ou um filho é criado em outro dispositivo, sua história é expandida para além do seu limite local, os dados armazenados localmente são, então, enviados a `EXEHDAbase` para ficarem visíveis aos demais entes. ISAMadapt incluiu a possibilidade de parte da História do ente ser privada: `private`. Neste caso, o ente obtém outro objeto da classe `History` somente armazena localmente suas tuplas. A semântica definida para leitura e escrita na História é implementada pelo componente EXEHDA-CC (YAMIN, 2004).

A aplicação pode operar sobre a História usando as operações definidas pela Hololinguagem (BARBOSA, 2002):

Escrita: `history ! <tupla>`

Leitura bloqueante: `history # <gabarito>`

Leitura não-bloqueante: `history #? <gabarito>`

Retirada bloqueante: `history . <gabarito>`

Retirada não-bloqueante: `history .? <gabarito>`

A aplicação refere-se à história privada com o qualificador `private`, e à história do pai com o qualificador `parent`. Para a operação de leitura é permitido o uso do qualificador `any`, que executa a operação na história local (`private`) e, se não encontrar a tupla procurada, executa a operação na história pública (`history`) do ente. Por definição do HoloParadigma, o ente somente tem acesso a sua história e a história de seu pai atual.

Se mais de uma tupla deve ser acessada, deve-se usar o conceito de lista: `tuple (tupla1, tupla2,...)`. O gabarito é uma tupla composta por strings, variáveis ou %variável, a qual designa a associação a ser feita.

O uso destes comandos no código ISAMadapt é traduzido para chamada às funções que disponibilizam a conexão com a parte do *middleware* (`lang.isamadapt.Runtime`) que implementa a coordenação entre os entes, conforme Quadro 5.3.

5.5.1.7 *Código Java não-padrão*

A geração de código Java não-padrão, como o Java Micro Edition – personalJava para entes que executam no Zaurus, deve estar dentro de blocos `native Java`. Por exemplo, no caso da aplicação-modelo `WalkEd` (ver Capítulo 6):

```
native Java  {*
    import org.jeode.pjava.*;
    ....
*}
```

5.5.1.8 *Variáveis de Ambiente*

As variáveis de ambiente são traduzidas para chamada ao método `getEnv`. Por exemplo: `myHost` é uma variável de ambiente que indica o equipamento preferencial do usuário, sendo traduzida para: `env.getEnv("myHost");`

Comandos ISAMadapt	Regras de Tradução

move to beingName
move to being: parent
move closerTo being:beingName
move to resource:resourceName
 	
move to resource:variable
 	
move to env:myHost
	isamadapt.Runtime.moveBeing ("to:being:nomeEnte");
	isamadapt.Runtime.moveBeing ("to:being:parent");
	isamadapt.Runtime.moveBeing ("closerTo:being:nomeEnte");
	isamadapt.Runtime.moveBeing ("to:resource:resourceName");
	// deverá ser atribuido à variable um comando
	// discovery anterior
	isamadapt.Runtime.moveBeing ("to:resource:variable");
	isamadapt.Runtime.moveBeing ("to:resource:env.myHost");

discovery var ("descrição_file") { …commandos };	ResourceDescriptor[] rd = IsamAdaptEngine.discoverResource ("descrição da busca de recurso", timeout); Symbol var = rd.Location;

reschedule()
reschedule ("all:")
reschedule("being:beingName");
	isamadapt.Runtime.reschedule(self);
	isamadapt.Runtime.reschedule("all:");
	isamadapt.Runtime.reschedule("being:Beingname");

reschedule("policy:all:",
"param:(nome,valor)");
	// comandos para alterar o arquivo policies.xml
	// durante a execução
	// na tag <scheduling> <policy> <allbeings> <paramNome>

reschedule("policy:being:Name",
"param:(nome,valor)");
	// comandos para alterar o arquivo policies.xml
	// durante a execução
	// na tag <scheduling> <policy> <being> <paramNome>

prefetch (lista-arq)	isamadapt.Runtime.prefetch(lista-arq);
install applic to target [start]	isamadapt.Runtime.install(applicName, target, true/false);
push source to target	isamadapt.Runtime.push(source, target);
disconnect	isamadapt.Runtime.disconnect ();
reconnect	isamadapt.Runtime.reconnect ();
kill (beingName)	isamadapt.Runtime.killBeing(beingName); //se lista vazia: self
exit	isamadapt.Runtime.exit();
native Java {* … *}	// copiar para o arquivo o bloco ente {* e *}
suspendNotify	adapt.IsamAdaptEngine.enableNotifications(F);"
resumeNotify	adapt.IsamAdaptEngine.enableNotifications(T);"
var:= getContext(nameElement)	Symbol var = adapt.IsamAadaptEngine.getContext(nameElement);
var:= checkContext(nameElement)	Symbol var = adapt.IsamAdaptEngine.checkContext(nameElement);

Quadro 5.2: Regras de Tradução dos Comandos ISAMadapt

Comando ISAMadapt	Regra de Tradução
Construtor do ente	History history = new History(“ ”); // cria história vazia
history ! var	history.out(var);

history (out) ! var	History parent = target.getParentHistory();
	parent.out(var);
	// target pode ser this ou adapter

private ! var	// na primeira vez
	History private = new History (“ “);
	private.out(var);

history # (%var, string) // block	Tuple pattern = new Tuple ( );
	pattern.add(null).add(string);
	var = history.in (pattern, timeout);

history #? (var, string, %var2) // no block	Tuple pattern = new Tuple ();
	pattern.add(var).add(string).add(null);
	var2 = history.in (pattern, 0);

history . (%var, string) // block	Tuple pattern = new Tuple ( ); Pattern.add(null).add(string); var = history.in_r (pattern, timeout);
history .? (var, string, %var2) // no block	Tuple pattern = new Tuple ( ); Pattern.add(null).add(string); var2 = target.in_r (pattern, 0);
parent ! list (tuple1, tuple2,...)	History parent = target.getParentHistory(); parent.out (tuple1); parent.out (tuple2); …

any #? (string, %var)	Tuple pattern = new Tuple ( );
	pattern.add(string).add(null);
	var = private.in (pattern, 0);
	if (var == null)
	  var = history.in (pattern, 0);

Quadro 5.3: Regras de Tradução para a História

6 ESTUDO DE CASO E EXEMPLOS DE APLICAÇÕES

Para validar a programação das construções propostas foi implementada uma aplicação bem conhecida, um editor de textos, chamado `WalkEd` (*Walking Editor*), cujas funcionalidades são conscientes do contexto em que a aplicação se encontra, executando num ambiente *pervasivo*. Neste Capítulo apresentam-se, também, outros exemplos de aplicações com comportamento *pervasivo*, que podem ser modeladas com o ISAMadapt.

6.1 Estudo de Caso – Editor *Pervasivo*

O objetivo desta aplicação é ilustrar algumas das construções ISAM e seu comportamento *pervasivo* capaz de adaptar-se às modificações em seu contexto de execução.

6.1.1 Definindo os Elementos de Contexto

A aplicação `WalkEd` é um protótipo simplificado cuja modelagem de suas funções considera a visão particular de contexto definido pelas características do dispositivo móvel em uso (capacidade e display); e ao estado da conexão de rede (`connected, disconnecting, disconnected`). Outro exemplo de elemento de contexto ao qual a aplicação poderia se adaptar é a quantidade de energia (bateria) do dispositivo.

6.1.2 Definindo as Estratégias de Adaptação

Para responder à variação do contexto, as estratégias de adaptação projetadas se referem a: alteração no código de alguns componentes da aplicação (carga dinâmica de código), alteração na interface apresentada ao usuário, habilitação/desabilitação de tarefas, e migração para nodos mais poderosos. Os comandos de adaptação a serem utilizados são `disconnect, reconnect, clone, move` e `discovery`.

6.1.3 Modelando a Solução

A aplicação `WalkEd` consiste em um editor de texto simples, sendo composto de três entes da aplicação e dois entes de sistema:

- `GUI`: implementa a interface gráfica do editor;
- `spell` e `dict`: implementam o serviço de verificação ortográfica;
- `print` e `file`: implementam os serviços de impressão e arquivo, respectivamente.

A Figura 6.1 mostra uma possível distribuição física da aplicação em diferentes momentos, conforme novos entes são criados.

6.1.4 Codificando a Aplicação

`WalkEd` inicia sua execução pela criação da interface gráfica do editor conforme ilustrado no fragmento de código da Figura 6.2. Na execução do comando `clone` tem lugar a primeira dimensão de adaptação da aplicação, a qual consiste na seleção implícita da implementação do ente `GUI` (`adapter being`) mais adequada ao dispositivo em que a aplicação está sendo disparada. O ente `GUI` é criado na inicialização da aplicação e retém, em sua história, o texto sendo editado. O ente é criado localmente, no nodo onde a aplicação foi disparada. O código da interface de edição (código do `adapter`) é carregado sob demanda com base na informação de contexto (tipo do dispositivo) disponibilizada pelo `ISAMcontextService`.

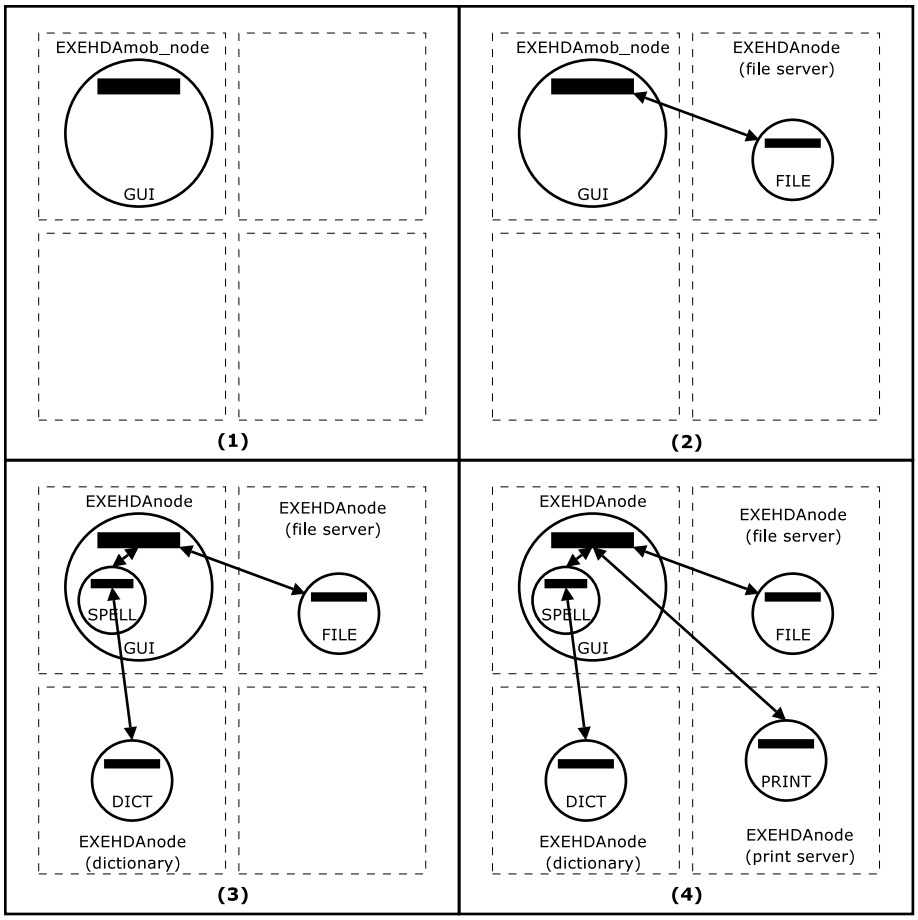


Figura 6.1: Entes da Aplicação `WalkEd`

No decorrer da execução, serviços como verificação ortográfica e impressão podem ser requisitados pelo usuário. Tais requisições disparam a criação de entes `spell`, `dict` e `print`, os quais podem vir a executar em dispositivos distintos daquele em que executa a interface gráfica. Disso decorre a segunda dimensão de adaptação: 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.

A terceira dimensão de adaptação da aplicação `WalkEd` está relacionada ao comportamento “siga-me” dos entes `spell` e `print` com relação ao 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 `spell` e `print` de forma a minimizar custos de comunicação respeitando, porém, suas dependências de acesso a recursos externos.

```
// file: walkEd.holo     ////////////////////
// WalkEd entry point
being holo
{
  holo ( ) {
     clone (GUI);
  }
}
// graphic interface adapted to display device
//
adaptive being GUI
  context deviceType::(desktop, PDA)
  context
      network::(disconnecting,connected)
{
   // shared methods  between interfaces
  openFile (filename) {
      clone (file(filename)), #fileId);
      contents=fileId.getContents();
      history ! tuple (filename, contents);
  }
  saveFile (filename) {  ….  }
  setActiveBufffer (bufferId) {  ….  }
  insertLine (pos, text) {  ….  }
  deleteLine (pos, n)  {  ….  }
  movecarret (line, pos) {  ….  }
  setSpellChecking ( ) {
      clone (spell);
  }
  gotoMyHost  {
     move to env:myHost;
  }
}
```

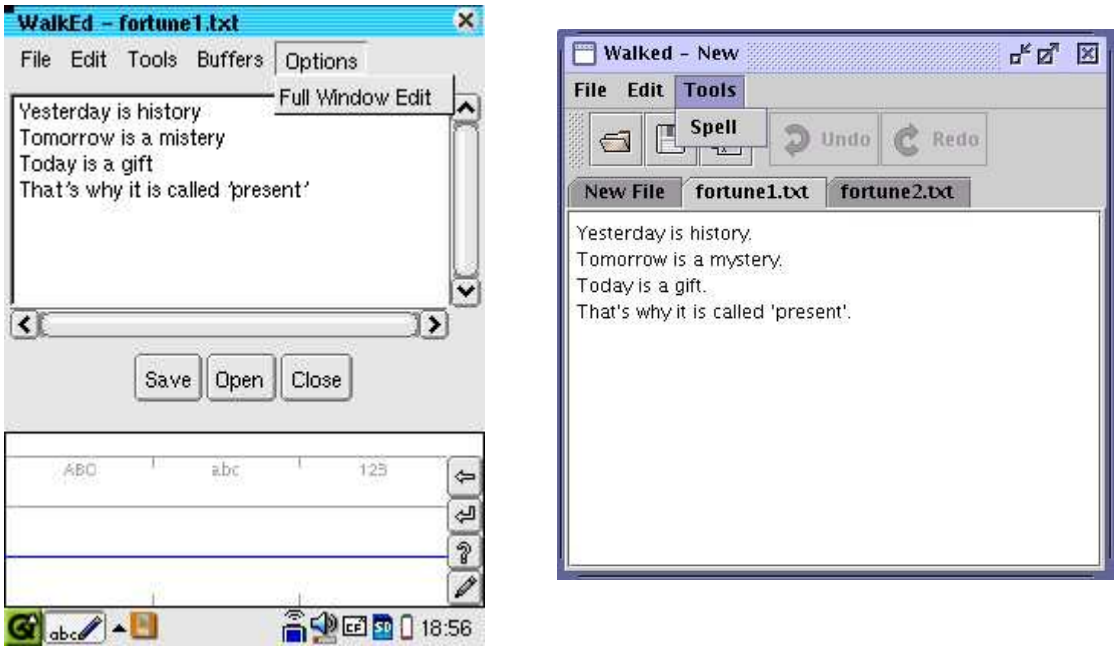
```
// file: interfacePDA.adp ////////////////////////////
 //
// WalkEd GUI implementation for PDA,
// version AWT/PersonalJava
//
// @description: target Sharp Zaurus.
// @context:  display :: PDA
//
 adapter being GuiBeing::GuiPDA( )
 {
     // Constructor
     GuiPDA( )   {
        native Java {*
         // use java.awt components to build WalkEd's GUI
         // and register menu callbacks
        *}
        onContext network::connected {
              enableSpellChecking( );
        }
     }
     disconnectedNetwork () {
           disconnect;   disableSpellChecking();
     }
     reconnectedNetwork () {
          reconnect;  enableSpellChecking;
      }
     startSpellCheck (line) { ….. }
     enableSpellChecking {
        native Java {* // update gui state
          …
        *}
        onContext network::disconnecting {
           disableSpellChecking ( );
        }
      }
     disableSpellChecking {
        native Java {* // update gui state
            …
         *}
        onContext network::connected       {
           enableSpellChecking ( );
         }
     }
           … // others methods
     }
```

Figura 6.2: Código dos entes `holo, GUI` e de um adaptador para o ente `GUI`

6.1.4.1 Adaptadores Alternativos para a Interface Gráfica

A interface gráfica adapta-se às capacidades de display do dispositivo em uso. As diferenças entre as interfaces gráficas nas duas plataformas, *desktop* e PDA, pode ser observada na Figura 6.3. Os adaptadores definem o código correspondente a cada estado do elemento de contexto display: PDA e *desktop*. Para tal, especializações do ente-base `GUI` são providas para as plataformas `PersonalJava`, disponibilizada em PDA's, e

`Java 2 Standard Edition`, disponível para *desktop*. Como a implementação da linguagem Holo não define uma API para interfaces gráficas, esses códigos usam a API Java diretamente através do comando `native Java`. Os métodos do ente usam os comandos de adaptação `clone, move.` A comunicação é realizada via *blackboard* (história do ente) através de operações para leitura (# e .) e escrita (!) de tuplas. As tuplas da história são compostas por *strings* que representam os parágrafos do texto em edição.



(a) (b)

Figura 6.3: Interfaces PDA and Desktop

A interface do `WalkEd` disponibiliza as opções mais comuns para edição de textos. **Open** cria um ente de sistema `file`, com o nome de arquivo dado pelo usuário, o qual cria a história com os registros do arquivo nomeado, e disponibiliza procedimentos para o envio de dados (`getContents`) e para gravação (`setContents`). O ente de sistema `file` é implementado também com comportamento adaptativo ao contexto, neste caso, `fileType`: texto, base de dados. O `WalkEd` permite abrir vários arquivos, porém com diferenças nas interfaces: *buffers* manipulados através do menu, para PDA, e arquivos manipulados em guias da interface, para *desktop/notebook*. O arquivo, se já existe, será buscado no espaço de nomes indicado (ava:, avu:, bda:), e gerado a história do ente contendo o texto já editado. **Save**, envia a história de volta para o ente `file`. **Print** cria um ente do sistema para tratamento da impressão, chamado `print`, o qual enviará o texto armazenado na História para a fila de impressão, conforme política associada ao recurso `printer`. Se o submenu `spellChecking` estiver habilitado (dispositivo não está no modo desconectado) e for chaveado para `on`, este criará um ente clone chamado `spell`. Os comandos de edição operam na história do ente (texto).

A qualquer momento, o usuário pode requisitar continuar a edição do texto em outro dispositivo do sistema. Isto é realizado através da ativação do submenu `goto`, após a qual o usuário é requisitado a selecionar o nodo destino (`myHost, myDesktop, myPDA,` etc). O sistema avalia esse alvo abstrato, com base nas preferências do usuário, armazenadas em seu ambiente virtual, para determinar o destino real da migração (processo de resolução de recursos do *middleware*). No destino, a aplicação

continua a executar. **Disconnect** avisa o sistema de execução para interromper a conexão com a rede, e desabilita funções de edição dependentes de informações remotas, como o `spellChecking`. **Reconnect** avisa o sistema de execução para conectar-se à rede, e reativa as funções desabilitadas.

6.1.4.2 O Comportamento Adaptativo da Correção Ortográfica

O ente `spell` é criado como filho do ente `GUI`, tipicamente co-localizado, e cria o ente `dict`, próximo ao recurso "dicionário", seguindo a política definida para a aplicação (Figura 6.7). O `spell` é sensível ao estado da conexão no nodo em que executa. Em seu modo de operação normal (`connected`), permanece lendo palavras em background a partir dos parágrafos modificados na interface e enviando essas palavras ao ente `dict`. A comunicação entre eles é através da história do ente (pai) `spell`. O ente `dict` consome a tupla `<"check", w1>` e retorna a tupla verificada com as sugestões de alterações: `<word, w1, ok>`
`<word, w1, reject, alter1, alter2, ... altern>`.

Quando o nodo torna-se desconectado, o `spell` não envia palavras ao `dict` a fim de reduzir o consumo de memória local, uma vez que as palavras não serão consumidas pelo ente `dict`. Deste modo, libera-se mais memória para ser usada pela edição do texto. Note que a troca do estado de conexão também altera o ente GUI que habilita/desabilita a opção de verificação ortográfica. O código do ente `spell` e dos adaptadores do método `bgSpellCheckWords` são resumidos na Figura 6.4. Note que a correção ortográfica somente tem efeito quando o dispositivo está conectado à rede.

```
// file: spellBeing.holo //////////////////////////
being SpellBeing
  context network
{
   adaptive  bgSpellCheckWords( );
   // constructor
   spell ( ) {
   // create the dictionary used for
  //  validate the words
      clone (dict, #dictionary);
      onContext network::connected {
           bgSpellCheckWords( );
      }
      ….
  }
}
```

```
//file: bgSpellCheckWordsV1.adp
// @context: newtork::connected
//
adapter
        Spell.bgSpellCheckWords:: v1 ( )
{ // wait for paragraph to be available
  //  in the parent's history
    parent # tuple ("textModified", %text);
    // send words to dict being
   history ! tuple ("check", language, palavras);
   // schedules next verification
  onContext network::connected {
      spellCheckWords();
  }
}
```

```
//file: bgSpellCheckWordsV2.adp
// @context: newtork::disconnected
//
adapter
        Spell.bgSpellCheckWords::v2 ( )
{
    // do nothing while disconnected
   //  to not fill up the device's memory
}
```

Figura 6.4: Código do ente `Spell` e dos adaptadores de `bgSpellCheckWords`

6.1.5 Executando a Aplicação

O código implementado foi testado usando-se dois tipos de equipamentos: *desktop* e o PDA, Sharp Zaurus 6500 com acesso sem fio. Nestes dispositivos, o EXEHDA foi previamente instalado. Para disparar a aplicação é usado o `ISAM Desktop` (Figura 6.5).

Após ser selecionada a aplicação, o Descritor de Disparo da Aplicação é carregado (`WalkEd.isam`), e o *middleware* passa a controlar a execução da aplicação, baseado nas políticas definidas para essa aplicação. O descritor é um documento XML, gerado durante a fase de desenvolvimento da aplicação, o qual agrupa uma série de meta-dados que provêem uma descrição abstrata da aplicação ISAM. O descritor usado pelo utilitário de disparo da aplicação (*launcher)* para o editor é apresentado na Figura 6.6.

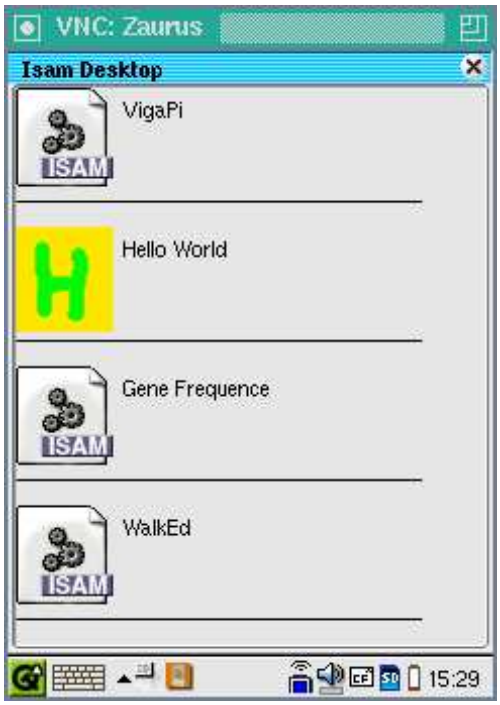


Figura 6.5: ISAM Desktop

```
<?xml version="1.0" encoding="UTF-8"?>
<isamapp spec="1.0" href="WalkEd.isam">
   <info>
      <title>Pervasive Editor </title>
      <vendor>ISAM team</vendor>
      <description>The Walking Editor </description>
      <icon href="WalkEd.png" />
   </info>
   <code>
      <main class="Main" />
      <jar href="bda:/WalkEd.jar"/>
   </code>
</isamapp>
```

Figura 6.6: O Descritor de Disparo da Aplicação `WalkEd`

Um exemplo das políticas de adaptação definidas para o `WalkEd`, escritas em XML, relativas aos comandos `reschedule, move, clone` (todos envolvidos com decisões de escalonamento) é apresentado na Figura 6.7. Essas políticas definem: (a) a

estratégia de escalonamento a adotar, selecionada pelas trocas de estado da conexão à rede; (b) o procedimento em caso de desconexão (emissão do comando `disconnect`): enviar os entes para nodos da rede fixa; (c) as âncoras para os entes `dict` e `spell`.

```
<scheduling>
    <strategy type="TiPS">
      <context v="network::connected">
          useCellScheduler=true
      </context>
      <context v="network::disconnected">
       useCellScheduler=false
      </context>
    </strategy>
  <policy>
    <allbeings>
     <context v=”network::connected”>
       <composition type=”replace”>
         <max index="CPU_POWER">  <!-- definição de parte da política dependente de contexto -->
       </composition>
     </context>
     <context v="network::disconnecting">
       <composition type="append">
         <rule n="moveToWired"/>
       </composition>
     </context>
    </allbeings>
  </policy>
  < policy composition=”replace”>
    <being name=”dict” />
    <anchor type=”avu:/dictionary.rsc” />
  </policy>
  <policy composition=”replace” >
     <being name=”Spell” />
     < anchor type=”being:GUI” />
  </policy>
  </policy>
 <rule name="moveToWired">
    <anchor sensor="NODE_TYPE" value="WIRED"/>
 </rule>
    <index name="CPU_POWER">
    <switch>
      <sensor name="HOST_BENCH[bogomips]" scale="10" />
      <sensor name="HOST_BENCH[linpack]" scale="4.75" />
      <composite type="sum">
       <sensor name="FREE_PHYS_MEM" />
       <sensor name="HOST_BENCH[scimark]" scale="4.75" />
      </composite>
    </switch>
  </index>
</scheduling>
```

Figura 6.7: Políticas Relativas às Decisões de Escalonamento

As políticas que orientam outras decisões, para definir as ações dos comandos, como a abordagem a adotar em caso de erro, são mostradas na Figura 6.8.

A Figura 6.9 esboça a seqüência de execução da aplicação pelo middleware após a instalação da aplicação (a). Note que à medida que a aplicação se desenvolve o middleware expande seu domínio para outros nodos (c).

```
<commands>
  <command name="move">
    <allbeings/>
    <attribute name="timeout" value="10s">
    <onerror>
      <retry n="5" delay="1s"/>
      <abort/>
    </onerror>
  </command>
  <command name="non-blocking read">
    <allbeings/>
    <attribute name="timeout" value="10ms"/>
  </commands>
</commands>
```

Figura 6.8: Políticas Definidas para Comandos do `WalkEd`

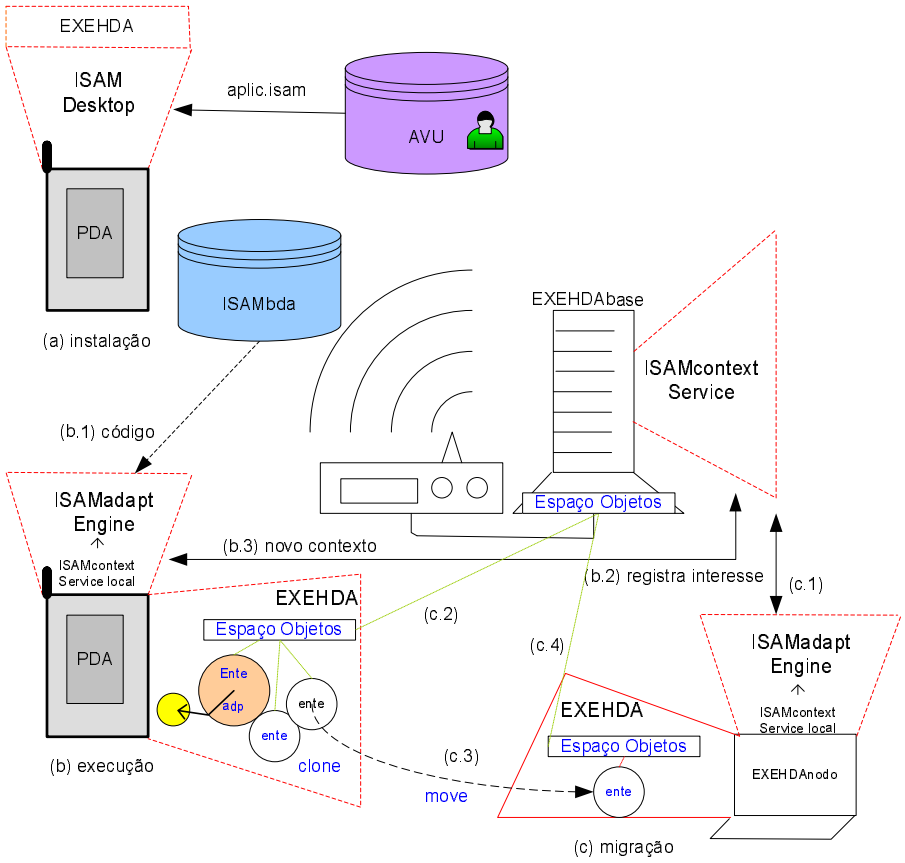


Figura 6.9: Esboço de Parte da Execução da Aplicação

Relativo a esta aplicação, pretende-se melhorar a interface do usuário e explorar outras estratégias adaptativas. Próximas etapas explorarão mais profundamente questões relativas à execução multicélula e multi-institucional, bem como estratégias de *caching* para melhorar o desempenho da aplicação.

6.2 Aplicações-Exemplo

Outras aplicações foram modeladas, mas não implementadas com ISAMadapt, e compõem o conjunto de aplicações-exemplo apresentado neste Capítulo. As aplicações

ilustram alguns domínios de aplicações onde a Computação *Pervasiva* pode ser inserida. O domínio de aplicações *Grid Computing* não é apresentado neste texto, pois é detalhado na referência (YAMIN, 2004).

6.2.1 Domínio CSCW: Reunião Virtual *Pervasiva*

Aplicações de *reunião virtual móvel* surgem da mobilidade de um grupo de usuários, e modelam, por exemplo, encontro de pesquisadores ou gerentes, onde cada integrante porta um computador com interface a uma rede sem fio. Usuários devem se comunicar para troca de documentos, informações e opiniões para realizar uma atividade compartilhada: reunião com pauta de discussão. A característica da reunião é assíncrona e com agenda pré-determinada pelo coordenador da reunião. Os participantes da reunião são previamente cadastrados, e lhes é atribuídos papéis pelo coordenador: relator, participante, ouvinte. Cada papel dá direito de acesso a determinadas funções e privilégios, sendo que o coordenador tem todos os direitos, e o ouvinte somente acessa as informações, mas não as gera (não tem direito à voz). Durante a reunião, o coordenador pode trocar os papéis dos participantes. A reunião inicia com uma proposta de pauta. Os participantes podem fornecer sugestões, argumentar com prós e contras. Transcorrido o tempo para discussões, será fornecida pelo relator a proposta de votação e estabelecido o prazo de votação. Cada participante vota uma única vez. A ata da reunião é emitida pelo relator, após concluída a votação.

6.2.1.1 O Software REUNI@o

Para exemplificar esse domínio de aplicações, foi escolhida uma aplicação existente, desenvolvida na Universidade Federal de Santa Maria (www.cpd.ufsm.br/~reuniao) (GOMES, 2001; MORO, 2001), a qual também exemplifica algumas das modificações para uma aplicação legada tornar-se *pervasiva*.

A aplicação distribuída Reuni@o é uma ferramenta de suporte a reuniões virtuais construída com base, e estendendo, o modelo IBIS (*Issue-based Information System* – Sistema de Informações baseado em Questões).

O modelo IBIS representa uma discussão como uma árvore, onde cada nodo é uma contribuição de um participante (Figura 6.10). As contribuições são classificadas em tipos:

- questão: problema, dúvida ou proposição que inicia um debate ou subtópico de um debate;
- idéia: propõe uma solução para a questão;
- argumento: dá sustentação ou contesta uma idéia.

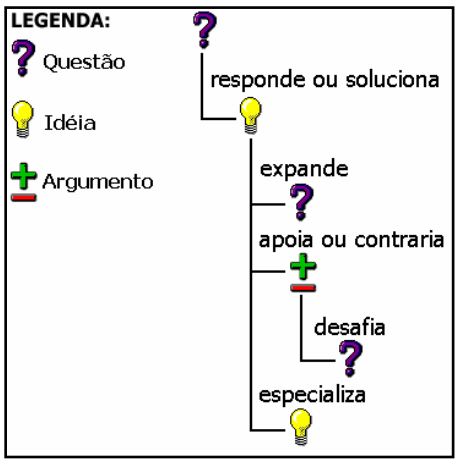


Figura 6.10: Relações entre os Tipos de Contribuição do Reuni@o

Anexos, como arquivos e URLs, podem ser adicionados para reforçar um ponto de vista (argumento pró ou contra). A partir desses conceitos é construída a árvore de discussão. O sistema Reuni@ao apresenta-se sob a forma de uma arquitetura distribuída, implementada em Java, com um servidor implementando comunicação com o(s) módulo(s) cliente(s) através de RMI. O módulo servidor é responsável pela implementação das funcionalidades de gerenciamento das contribuições. O módulo cliente tem a responsabilidade de implementar a camada de apresentação da aplicação, armazenamento local das informações e comunicação com o servidor utilizando os serviços oferecidos por este (GOMES, 2001). A Figura 6.11 mostra a interface do módulo cliente de referência.

Figura 6.11: Interface Cliente no Reuni@o

6.2.1.2 Adaptação do Reuni@o ao Ambiente Móvel

De acordo com Gomes (2001), um dos requisitos da aplicação do sistema Reuni@o é ter facilidade para incorporação de novas interfaces de usuários e extensão das funcionalidades da ferramenta. Deste modo, iniciou-se a criação de um protótipo da versão cliente do sistema Reuni@o para dispositivos móveis, REUNI@O^{Móvel}, como telefones celulares inteligentes (CHIELE, 2003). A Figura 6.12 mostra uma das telas do protótipo em um celular, implementado usando J2ME/MIDP. Porém, esse protótipo ainda não é adaptativo ao contexto. A dificuldade da implementação adaptativa, segundo o modelo ISAMadapt, está em que o perfil MIDP (ver Apêndice A) não permite serialização de objetos, nem possui comunicação remota via RMI. Isso exige uma mudança significativa na modelagem dos entes que executam no celular, em relação aos que executam no *desktop* e no PDA. As principais adaptações previstas para serem implementadas são apresentadas a seguir.

6.2.1.3 Definindo os Elementos de Contexto

A aplicação proposta é um protótipo simplificado, com algumas funções da Reunião Virtual, modelada considerando a visão particular de contexto definido pelas características do **dispositivo móvel em uso** (`display`) e ao **tipo de conexão de rede** utilizado (`wired, wireless, disconnected`).

Os **papéis** definidos para os participantes constituem outro elemento de contexto, que pode ser trocado dinamicamente a partir da observação do perfil de participação na reunião (elemento de contexto dinâmico e lógico). Regras dinâmicas de uso definem o perfil. Estas podem ser adicionadas, removidas ou modificadas, são dependentes do contexto, e implementam obrigações, permissões e proibições do papel.

Figura 6.12: Menu do REUNI@O[Móvel]

6.2.1.4 Definindo as Estratégias de Adaptação

A interface do software apresentada é dependente do papel do participante da reunião, além do ajuste correspondente ao tipo de dispositivo sendo utilizado. Por exemplo, se o usuário tem o papel `administrador`, a tag `Administração` aparecerá na interface, permitindo criar reuniões e sessões, definir o coordenador da reunião, etc. O papel `coordenador` tem direito à tag `Usuários` para definir os demais participantes da reunião, seus papéis (que podem ser modificados dinamicamente), permissões e obrigações dos papéis, e a tag `Reunião` para controlar o desenrolar da reunião: criar pauta, iniciar trabalho do relator, etc. Se o participante tem somente o papel de `ouvinte`, é desativada a opção `Envio` do menu.

A fidelidade de saída é usada para ajuste aos recursos da rede: alta fidelidade quando os recursos são plenos, e baixa fidelidade quando são escassos. Estratégias de adaptação usam migração (`move`), código sob demanda (implícito), disseminação de dados (`push`), ativação/desativação de atividades (papéis), *prefetching* de anexos e filtragem

de dados para ajuste da capacidade dos recursos, e sincronização de *buffers* locais para tratamento da desconexão. Uma variedade de filtros (compressão, redução de imagem, conversão de formato) poderá será inserida entre o cliente e o servidor para reduzir a necessidade de recursos (ver item 6.2.3: multimídia adaptativa).

6.2.1.5 Modelando a Solução

Muitas aplicações que dependem da capacidade da rede tendem a tratar PDAs como clientes leves onde atuam somente como dispositivos de entrada/saída. No reuni@ao, os clientes têm embutidos outras atividades, como armazenamento local quando desconectados, e *prefetching* de anexos.

A aplicação tem três grandes processos: (i) login, (ii) gerar informações e (iii) receber informações dos demais participantes. As informações são textos em sua maioria; anexos e URLs compõem informações usadas como reforço de argumentação. A adaptação poderá ocorrer, nesses processos, da seguinte forma:

- login - a aplicação inicia adaptando a interface ao dispositivo corrente, pedindo a este para selecionar a reunião da qual participará, e solicitando a identificação do participante para certificação (id, senha). Estes dados são enviados ao servidor que certifica o usuário como integrante da reunião. A informação de tipo de conexão e papel do participante é utilizada para carregar os demais componentes que comporão a interface da aplicação para o dispositivo;
- emissão – é relativa ao envio das contribuições e dos anexos, que considera o tipo de conexão corrente: `forte/wired` – alta fidelidade; `fraca/wireless` - baixa fidelidade, envio postergado (em partes); `desconectado` – *buferização* local para sincronização futura;
- recepção – é relativa ao recebimento das contribuições e dos anexos, que considera: tipo de display (`pleno=notebook, restrito=PDA, insuficiente=celular`) e conexão corrente (`forte=wired, fraca=wireless, desconectada`). Display pleno: toda a árvore de discussão pode ser carregada, se com conexão forte; se a conexão está fraca, somente a parte não lida pelo participante é carregada, as demais são carregadas por demanda (solicitação do participante); desconectada, é perguntado ao usuário se este deseja conectar-se para a recepção de mensagens não lidas. Se as informações não lidas possuem anexo, a adaptação a ser realizada é: sob conexão forte, carga e disparo do aplicativo correspondente para leitura; sob conexão fraca, carga e disparo do aplicativo para leitura, em ordem de prioridade definida pelo participante; desconectado, mensagem ao usuário para conectar e receber indicativos de anexos (nomes e URLs), e carga destes sob demanda (pedido do usuário).

Em caso de desconexão (a informação `conectado/desconectado` está sempre presente na barra de status da ferramenta), é requerido pela aplicação que esta deve permanecer funcional e permitir sincronização tão logo a rede esteja disponível. Para tanto, é necessário manter o armazenamento local de informações, e prover mecanismos de sincronização na reconexão. No modo conectado (`forte` ou `fraco`), um ente na rede fixa envia as novas informações para o participante, conforme estas são geradas ou um intervalo de tempo foi definido para o recebimento das novas informações.

6.2.1.6 *Codificando a Aplicação*

As Figuras 6.13, 6.14, 6.15 e 6.16 apresentam um esboço de código para os entes e adaptadores dessa aplicação.

```
being holo
{
   holo ( ) {
     clone (login);
    //  login e senha para certificação do participante
     clone (contrib); // abre a base de dados
   }
}
```

```
being contrib
{ //
   discovery BD (ava:/BD) {
       clone (file(BD), #bd);
       history ! tuple (“BD”, bd.getContents());
   }
   while true {
     sync onContext storeBD {
       history # tuple (“BD”, bd.setContents());
     }
   }
```

Figura 6.13: Esboço dos Entes `holo` e `contrib`

```
adaptive being login
   context  deviceType::(cellular, PDA, desktop);
   context  role::(administrator, coordenator,
                   relator, member, listener)
{  adaptive  setAttachRead(name, type)
       context typeFileConnection;

   setTree ( ) {
      clone (tree, #tree);
   }
   setPush() {
       clone (pushBeing);
   }
   setEdit () {
     // unavailable for listener
     clone (editBeing);
   }
   setSend ( ) {
        clone (sendBeing);
   }
    setAttachRead (name, type) {
         // display attach request
    }
   ….
}
```

```
////////// PDAlogin.adp ///////////////
//@context: deviceType
adapter being login::PDAlogin
{  context connection::(disconnected, wired, wireless);
   context role::(administrator, coordinator, relator,
                  member, listener);

   PDAlogin (idUser) {
     identifyUser(iduser);
     native Java {*
         // code j2ME/personalJava for Zaurus
         //  use the awt library for interface implementation
         // show and select the meeting
         // show basic interface
     *}
     // add new components
     onContext  role::coordenator {
          native Java {* // add tag about coordinator role
          *}
     }
     onContext role::administrator { …}
     …..
     onContext network::disconnected {
         native Java {*
           // alter the status bar and active the reconnect button
         *}
     }
     onContext network::(wireless, wired) {
        native Java {*
          // alter the status bar and deactive the reconnect button
        *}
     }
   }
   identifyUser (usr) {
        …
  }
 …..
}
```

```
////////// desktopLogin.adp  ////////////////
//@context: deviceType::desktop
adapter being login::DesktopLogin
{
   desktopLogin (user) {
       native Java {*    // import code of the Reuni@o
       *}
       setTree(); …
   }
}
```

Figura 6.14: Esboço dos Entes de Interface e Adaptadores

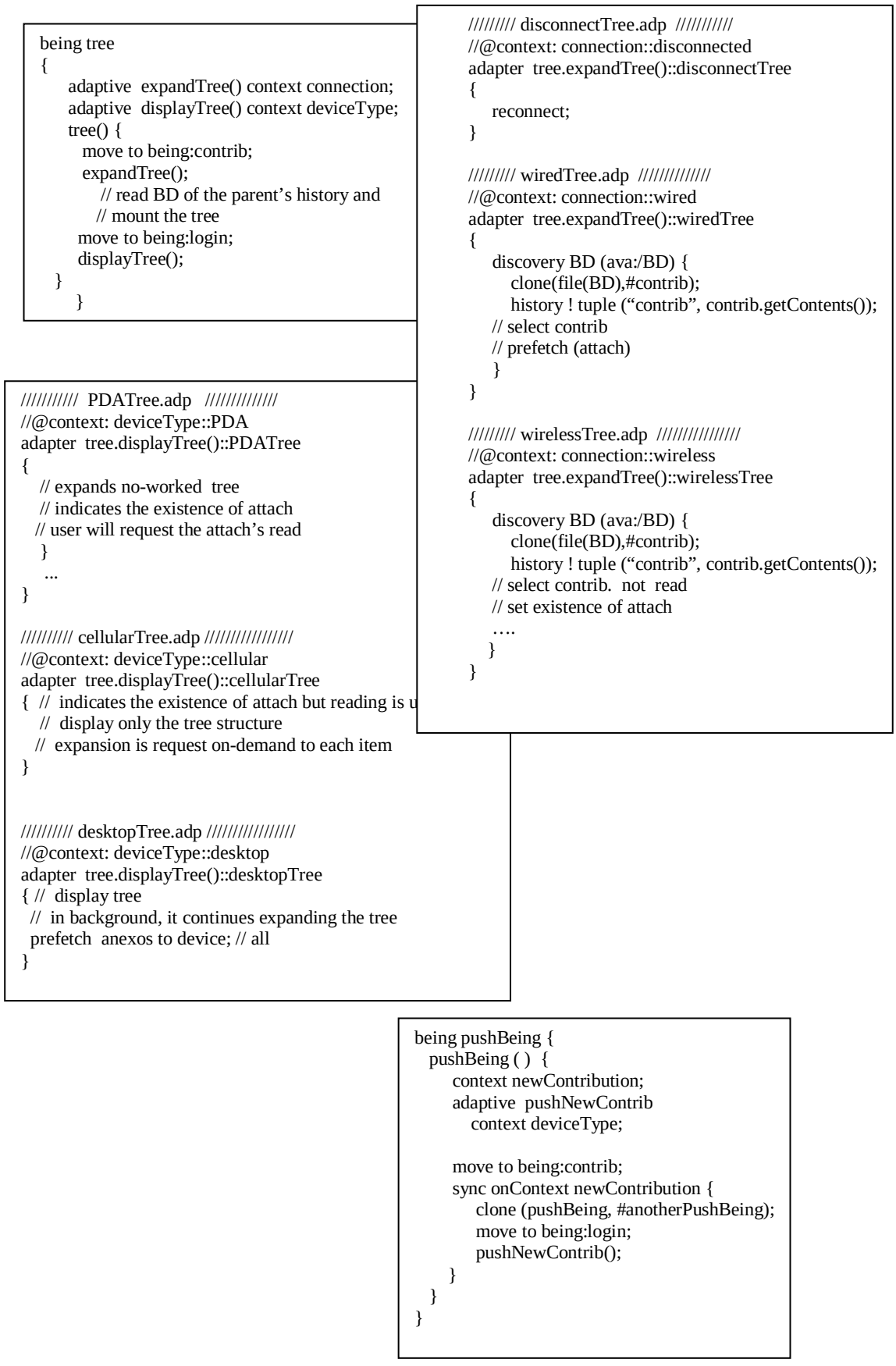


Figura 6.15: Esboço dos Adaptadores para Recebimento de Contribuições

```
being sendBeing
{  // shared method
   adaptive  sendTo()
       context deviceType::(cellular, PDA, desktop);
  sendBeing() {
    move to being:editBeing;
    copyHistory();
     // copy the history of parent:editBeing
    sendTo();
  }
}
```

```
//////// wirelessSend.adp  ///////////////
//@context: connection::wireless
adapter  sendBeing.sendTo()::wirelessSend
{  // mount a block of history tuple each time
   while ("private # tuple ("block", %content) do {
      push content  to being:contrib;
      sleep (100);
   }
   clone (sendBeing);
}
```

```
//////// disconnectedSend.adp ///////////
//@context: connection::disconnected
adapter  sendBeing.sendTo()::disconnectedSend
{
   reconnect;
}

//////// wiredSend.adp ////////////
//@context: connection::wired
adapter  sendBeing.sendTo()::wiredSend
{
  move to being:contrib;
  parent ! copyAll;
  // arrive of this being trigger the alter of context: new Contribution
  clone (sendBeing); // create another being – actor style
}
```

Figura 6.16: Esboço dos Adaptadores para Emissão de Contribuições

6.2.2 Domínio Agentes Adaptativos: Avaliação à Distância

O uso de agentes móveis tem sido proposto para modelar aplicações que executam em ambientes móveis/Internet. Em geral, as implementações de agentes móveis não disponibilizam adaptação ao contexto, pois foram projetados para executarem no ambiente distribuído tradicional, onde este comportamento não é tão relevante quanto no *pervasivo*. Um **ente** pode modelar o comportamento de agentes móveis, e tirar vantagem da adaptação ao contexto, além de não necessitar carregar seus dados e código junto a si. Neste sentido, o exemplo de aplicação a seguir é relativo a um sistema de avaliação à distância de estudantes conectados por diferentes tipos de redes, e usando diferentes dispositivos (rede *pervasiva*). A característica básica dessa aplicação é seu **comportamento pró-ativo**, determinando as atividades dos usuários.

6.2.2.1 Definindo os Elementos de Contexto

O controle do processo de avaliação é baseado em um cronograma que estabelece datas para a finalização das tarefas envolvidas nesse processo. O cronograma é o algoritmo base para a definição de um contexto do tipo lógico, definido pela aplicação: cada data e hora estabelecida no cronograma define um novo estado do contexto `cronogramTask`. Outros elementos de contexto ao qual a aplicação é sensível são: o estado de conexão à rede (`wired, wireless, disconnecting`), e o tipo de dispositivo (`PDA, cellular, desktop`). As preferências dos examinadores também podem ser usadas como elemento de contexto.

6.2.2.2 Definindo as Estratégias de Adaptação

Como esta é uma aplicação que gerencia as tarefas a serem executadas pelos usuários, os comandos `install`, `push` e `move` são usados para expressar as estratégias de adaptação. O comportamento adaptativo do ente de sistema `file` também é importante nessa aplicação.

6.2.2.3 Modelando a Aplicação

A aplicação de avaliação compreende três estágios, baseados em um cronograma: (i) geração do exame; (ii) distribuição e resolução do teste; (iii) compilação dos resultados.

A aplicação `Avalia`, executada pelo usuário `secretary`, inicia com o ente `holo` controlando o cronograma e a geração, no tempo previsto, dos entes que executarão os estágios da avaliação. O estágio de geração do exame é realizado por questões geradas por diferentes examinadores que estão em diferentes localizações. No tempo previsto pelo cronograma é instalada a aplicação `createQuestions` que permite ao examinador gerar suas questões. Seguindo o cronograma, o ente `fetch` é despachado para buscar as questões de todos os professores. O itinerário deste ente é baseado em informações de contexto relativo ao perfil do professor e questões de conectividade à rede. Quando todas as questões foram recolhidas, o ente `assembler` gera o exame, com base numa escolha aleatória das questões parciais enviadas pelos examinadores, armazenando-a. O estágio de distribuição e resolução do teste é iniciado na data e hora prevista no cronograma, as questões são enviadas ao dispositivo dos estudantes que realizarão a prova, permitindo que estes estejam desconectados. Caso ocorra erro nesta operação, a secretaria é avisada. O ente `testApplic` irá controlar o horário de realização de cada questão, obtendo a resposta após transcorrido o prazo, e apresentando a próxima questão ao estudante. A resposta obtida é armazenada na `history`, junto com os dados de horários de realização da questão, informações que serão usadas em estudos estatísticos. Quando o teste for concluído, o ente `testApplic` migra para o ente `holo`. O estágio de compilação dos resultados é realizado de forma distribuída por tipos de questões. Questões dissertativas são enviadas aos respectivos examinadores pelo ente `answer`, enquanto que questões objetivas são processadas pelo ente `score`, criado junto à base de respostas da aplicação. Após o ente `answer` ter recolhido a nota relativa às questões dissertativas, este migra para o ente `score` e processa o resultado final do estudante. O ente `publisher` divulga os resultados aos examinadores e aos estudantes, emitindo um relatório para a secretaria do curso.

6.2.2.4 Codificando a Aplicação

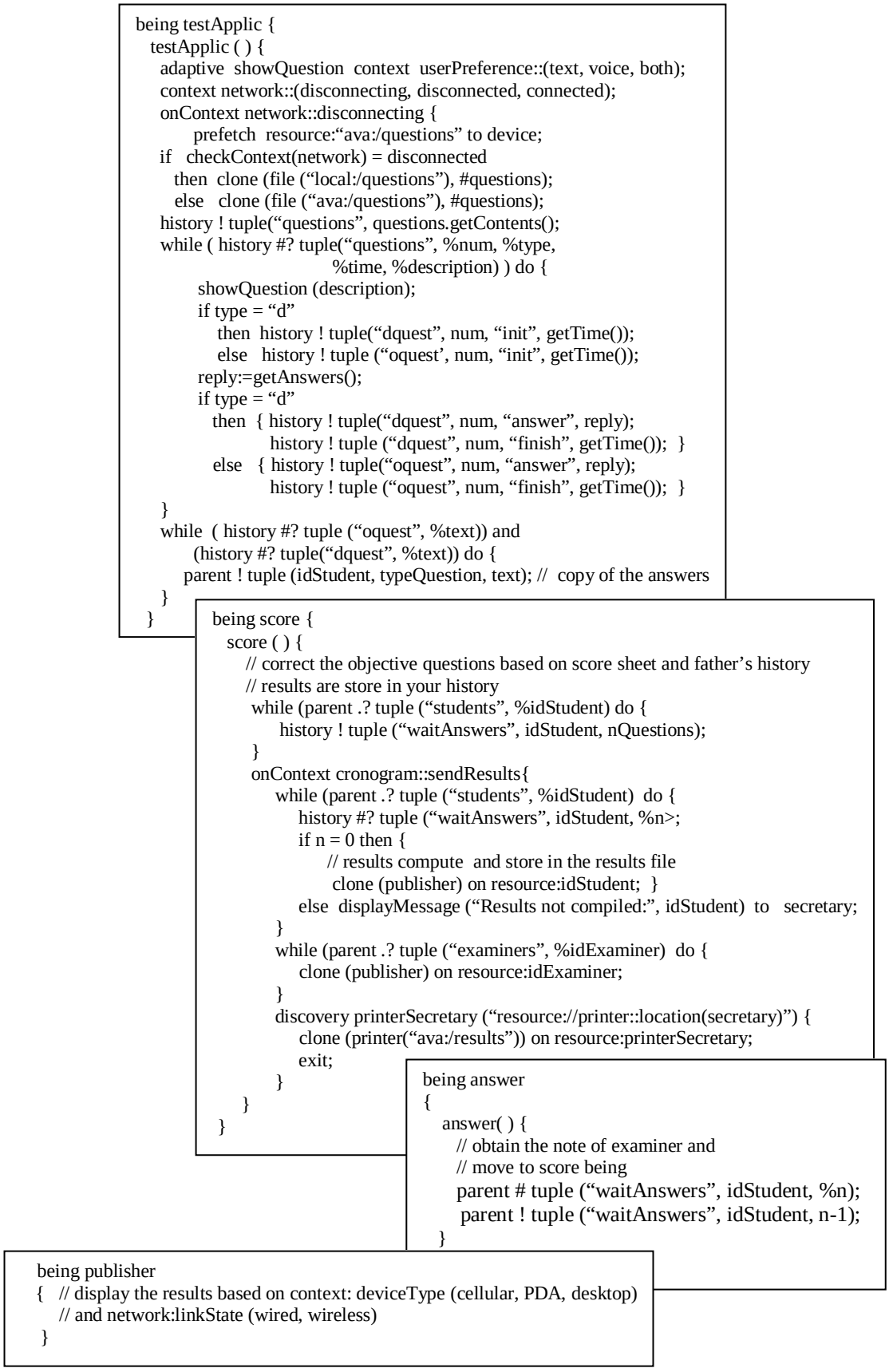
Um esboço de codificação desta aplicação é dado nas Figuras 6.17 e 6.18.

```
being holo   {
  holo()   {
      context cronogram;
      clone (file("examiners"), #fExaminers);  private ! tuple ("examiners", fExaminers.getContents());
      clone (file ("students"), #fStudents);       private ! tuple ("students", fStudents.getContents());
     onContext cronogram::createQuestions {
           while  (private .? tuple ("examiners", %idExaminer)  do {
              install createQuestions at  idExaminer [start];
           }
       }
     onContext cronogram::fetchQuestions {
        while (private .? tuple ("examiners", %idExaminer) do {
            clone (fetch, #idFetch)  on resource:idExaminer;
        }
     }
    onContext cronogram::allQuestions {
         clone (assembler, #createTest) anchorOn  resource:"ava:/baseQuestions";
        // baseQuestions.rsc  describes, in xml format,  the resource
        //  assembler being create the test's questions and store in the questions file.
    }
    onContext cronogram::initTest{
        for (private .? tuple ("students", %idStudent)  do {
            onError {
               clone (testApplic, #idTest) on resource:idStudent;
            }  displayMessage ("Student not connected:", idStudent)  to secretary;
        }
    }
    onContext cronogram::compilResults {
               clone (score, #idscore);
               // send the dissertation questions to examiners
               while (private  .? tuple ("examiners", %idExaminers) do {
                   // create file with selection of the questions  to each examiner
                  push  idExaminerQuestions to  idExaminers;  //avu of the examiners
                  install  correctAppl  at  idExaminers;
                  onContext userConnected (idExaminers) {
                        clone (answer);
                  }
               }
      }
      ….
}
```

```
being fetch
{  context network;
   fetch()   {    // send the examiner's questions
      onContext network::connected {
          push  myQuestions  to resource:"ava:/baseQuestions";
         // retry  is the default policy
     }
   }
}
```

Figura 6.17: Esboço do Código dos Entes `holo, fetch`.

Note que a aplicação que não inicia imediatamente (`install`), porque o usuário está desconectado, é instalada no ambiente virtual do usuário, pois este poderá acessá-la quando se conectar, em qualquer lugar. O comando `exit` é necessário para encerrar a aplicação, no caso de alguns entes ficarem pendentes.



```
being testApplic {
  testApplic ( ) {
    adaptive showQuestion context userPreference::(text, voice, both);
    context network::(disconnecting, disconnected, connected);
    onContext network::disconnecting {
         prefetch resource:“ava:/questions” to device;
    if checkContext(network) = disconnected
      then clone (file (“local:/questions”), #questions);
      else clone (file (“ava:/questions”), #questions);
    history ! tuple(“questions”, questions.getContents();
    while ( history #? tuple(“questions”, %num, %type,
                              %time, %description) ) do {
          showQuestion (description);
          if type = “d”
             then history ! tuple(“dquest”, num, “init”, getTime());
             else history ! tuple (“oquest’, num, “init”, getTime());
          reply:=getAnswers();
          if type = “d”
            then { history ! tuple(“dquest”, num, “answer”, reply);
                   history ! tuple (“dquest”, num, “finish”, getTime()); }
            else { history ! tuple(“oquest”, num, “answer”, reply);
                   history ! tuple (“oquest”, num, “finish”, getTime()); }
    }
    while ( history #? tuple (“oquest”, %text)) and
          (history #? tuple(“dquest”, %text)) do {
         parent ! tuple (idStudent, typeQuestion, text); // copy of the answers
    }
  }
```

```
being score {
  score ( ) {
     // correct the objective questions based on score sheet and father’s history
     // results are store in your history
      while (parent .? tuple (“students”, %idStudent) do {
           history ! tuple (“waitAnswers”, idStudent, nQuestions);
      }
      onContext cronogram::sendResults{
          while (parent .? tuple (“students”, %idStudent) do {
              history #? tuple (“waitAnswers”, idStudent, %n>;
              if n = 0 then {
                   // results compute and store in the results file
                    clone (publisher) on resource:idStudent; }
              else displayMessage (“Results not compiled:”, idStudent) to secretary;
          }
          while (parent .? tuple (“examiners”, %idExaminer) do {
              clone (publisher) on resource:idExaminer;
          }
          discovery printerSecretary (“resource://printer::location(secretary)”) {
              clone (printer(“ava:/results”)) on resource:printerSecretary;
              exit;
          }
     }
  }
```

```
being answer
{
   answer( ) {
     // obtain the note of examiner and
     // move to score being
     parent # tuple (“waitAnswers”, idStudent, %n);
      parent ! tuple (“waitAnswers”, idStudent, n-1);
   }
```

```
being publisher
{  // display the results based on context: deviceType (cellular, PDA, desktop)
   // and network:linkState (wired, wireless)
 }
```

Figura 6.18: Esboço do Código dos Entes `testApplic`, `score`, `answer`, `publisher`

Exemplos de políticas que podem direcionar essa aplicação são apresentados na Figura 6.19.

```
<policy composition="replace">
  <being name="assembler"/>
  <anchor type="ava:/baseQuestions" />
</policy>
<policy composition="replace">
  <being name="score"/>
  <anchor type="ava:/oQuestions" />
</policy>
…
```

```
<commands>
  <command name="install" >
     <allbeings/>
     <attribute  name="timeout" value="10s" />
     <onerror>
        <retry n="24" delay="60min" />
        <message to="ava:/secretary.rsc"/>
        <abort />
     </onerror>
  </command>
  <command name="clone" >
     <allbeings/>
     <attribute  name="timeout" value="1s" />
     <onerror>
        <retry n="10" delay="10s" />
        <abort />
     </onerror>
  </command>
…..
```

Figura 6.19: Exemplos de Políticas para o Avalia

6.2.3 Domínio Multimídia Adaptativa: Adaptação de Conteúdo

Apesar de os novos dispositivos móveis terem aumentado a capacidade de processamento, armazenamento e display, ainda cuidados apropriados devem ser tomados em relação a condições da mídia e do dispositivo que podem alterar o resultado final, devidos a fatores como resolução, banda, CPU disponível, consumo de energia e capacidade de display.

Adaptação de conteúdo refere-se à modificação da representação da informação a fim de alcançar as capacidades de tratamento da mídia do dispositivo e as restrições de transmissão impostas pela rede sem fio (KARUNANIDHI, 2002). Tais modificações podem incluir: transcodificação de formato (por exemplo: jpg, bmp, wbmp), escala de imagens, vídeo e som, conversão de mídia (por exemplo: texto para fala), omissão ou substituição de partes de um documento (por exemplo: imagem substituída por descrição textual), ou fragmentação de documentos.

Muitos esforços têm tratado dos problemas relativos à adaptação de conteúdo, principalmente em aplicações *Web*. Diferentes abordagens de adaptação de conteúdo podem ser identificadas:

a) adaptação do lado do servidor, onde o servidor fornece o documento adaptado. A adaptação pode ser realizada dinamicamente sob demanda, ou o servidor pode ter um repositório de documentos adaptados previamente (armazenados em diversos formatos);
b) adaptação baseada em *proxy*, os documentos são fornecidos pelo servidor em uma representação genérica, o *proxy* realiza a adaptação sob demanda. O *proxy* pode residir no lado do cliente (móvel), no nodo do servidor, ou em nodos intermediários;
c) caminhos de adaptação, onde um *pipeline* de *proxys* realiza uma sequência de adaptações etapa-por-etapa.

Normalmente, a adaptação de conteúdo é transparente ao usuário (sem interferência deste), pré-determinada com base em contextos fixados pelos sistemas. No entanto, o interesse do ISAMadapt é oferecer possibilidades às aplicações para definirem o comportamento adaptativo da mídia com base em contextos elaborados por estas.

A funcionalidade da adaptação de conteúdo pode ser modelada com a definição de **filtros**, disponibilizados através de entes do sistema (`system beings`), que poderão ser **reutilizados** no código das aplicações. Quando a aplicação requisita uma informação multimídia, esta requisição é codificada através de um ente de sistema `filter`, que implementa a adaptação conforme políticas definidas para esse ente. Elementos de contexto, que definem o tipo de adaptação de conteúdo (filtro), incluem preferência do usuário, capacidade do dispositivo do usuário ou condições da rede. Se o dispositivo pode tratar com diversas representações da mídia, uma prioridade (definida no perfil do usuário) define a seleção da representação a ser usada. O `avu` pode armazenar um documento totalmente adaptado, parcialmente adaptado ou com sua representação genérica. A localização do `ente` que implementa a adaptação de conteúdo pode ser dinâmica, dependendo das condições do contexto, e é definida na política codificada para a aplicação, ou na clonagem do ente `filter`. Um esboço dessa solução é apresentado na Figura 6.20, onde o método adaptativo `displayImage` utiliza filtros em sua codificação.

Diferentes estratégias de adaptação poderão ser projetadas, tais como: quando um vídeo trafega em banda baixa e a qualidade do áudio é importante, esta pode ser priorizada, e o foco da adaptação recai sobre a imagem. Essa solução considera que a informação multimídia será representada genericamente por meta-dados descrevendo-a em XML, de forma a simplificar a aplicação de filtros ou conversores que realizam a adaptação de conteúdo. O padrão MPEG-7 (*Multimedia Content Description Interface*) e muitos algoritmos de filtragem e conversão disponíveis na literatura podem ser utilizados para a implementação desses adaptadores (LIE, 2000).

Novas aplicações que usam dados multimídia, como o Reuni@o, podem definir no código métodos adaptativos e associar a eles adaptadores, já codificados em outra aplicação, com a funcionalidade adaptativa desejada (reusabilidade) e o elemento de contexto ao qual são sensíveis (flexibilidade). Por exemplo, se for dado ao usuário a possibilidade de ver ou ouvir os dados da reunião, a aplicação define o contexto, o nome do método adaptativo e associa ao estado do contexto o adaptador de método com o efeito correspondente, usando o *ISAMadapt IDE/menu adapters*. Novos adaptadores podem ser criados e disponibilizados (extensibilidade).

A adaptação de conteúdo é um comportamento também associado ao comando `prefetch` e comando `push`, ainda não implementados no ISAMadapt, os quais poderão indicar o filtro a usar para execução do comando.

```
//@context:  deviceType::PDA
//@description:  adaptive multimedia content
adapter
   myBeing.displayImage(fileDescriptor)::PDA
{
  clone (filter(grayScale, fileDescriptor),
       #grayImage) on resource:server;
  //filter  runs the content adaptation, and
  // policies indicate for store image information
  // in father's history
  display (Image);
 //  this method read the being's history and
 //  show the image
…  }
```

```
adapter
myBeing.displayImage(fileDescriptor)::cellular
{
   clone (filter(sizeReduce, fileDescriptor))
        on resource:server;
   // policies indicate store image information
   //  in tempFile
   clone (filter(grayscale, tempFile) on
        resource:EXEHDAbase;
   display (Image);
   …
}
```

Figura 6.20: Esboço de Adaptadores de Método usando Filtros

6.2.4 Domínio *Location-aware*: Fiscalização

Uma nova classe de aplicações, conhecida como sensível à localização (*location-aware applications*) está emergindo. Aplicações dessa classe obtêm informações sobre a posição corrente do usuário, e alteram o conteúdo e/ou recurso em uso baseados nesta informação. Existem diferentes tipos de aplicações que podem tratar com consciência da localização (IEEE Computer, 2001). Numa visão global, podem abordar sistemas de informações relativos a cidades, tráfego e emergências; numa visão mais local, podem abordar sistemas de informações sobre museus, pontos turísticos.

Uma aplicação, atualmente sendo desenvolvida com uma visão nômade (não adaptativa) pelo CPD-UFSM, é a de automação do setor de fiscalização e vistoria das prefeituras. Este setor inclui vigilância sanitária, fiscalização tributária, fiscalização de transportes, comércio e meio ambiente. Uma proposta de solução *pervasiva*, considerando os conceitos do ISAMadapt, pode ser projetada para explorar a mobilidade física do usuário, a qual altera a funcionalidade da aplicação conforme altera o contexto do usuário que a está executando.

A natureza do trabalho de fiscalização exige que os fiscais se desloquem para diferentes tipos de estabelecimentos (mobilidade). Cada tipo de negócio tem características e legislação própria e, conseqüentemente, diferentes procedimentos de fiscalização (adaptação). Os fiscais devem ir ao local de fiscalização, com informações sobre legislação, sobre o estabelecimento e ocorrências anteriores; preencher os autos de infração, coletar comprovantes (fotos, documentos) e enviar as informações para o centro de operações. A solução proposta equipa os fiscais com dispositivos móveis com câmeras e impressoras acopladas, conectados a uma rede sem fio, dando-lhes mobilidade de ação. A aplicação é adaptativa à localização (estabelecimento em fiscalização), onde varia a legislação, os documentos, as informações e ocorrências passadas. Essa solução permite a flexibilidade do sistema com o acesso a informações dependentes da localização (estabelecimento) em que o fiscal faz o seu trabalho no momento. O fiscal poderá se deslocar para qualquer estabelecimento sem prévia instalação de documentos e software para aquele tipo de fiscalização (solução atual sendo implementada impõe esse procedimento).

Nesta proposta, o PDA do fiscal funciona como um portal, onde dados e código são dependentes do contexto tipo de estabelecimento. O sistema fará a instalação automática, usando a tecnologia *push* de disseminação de dados, dos documentos, dados e código necessários, com base na informação de localização. Fotos e outros documentos poderão ser digitalizados pelo fiscal com o uso de câmeras acopladas ao dispositivo portátil. O retorno de informações para o centro de operações não mais exigirá a presença do fiscal para sincronização dos dados, este as enviará usando o sistema de comunicação móvel dos equipamentos. Multas, ou outros documentos, poderão ser gerados pelo fiscal imediatamente, usando a impressora portátil acoplada ao dispositivo móvel. Este procedimento reduz o tempo do processo de fiscalização, dá mais transparência ao processo e pode ajudar no combate às fraudes.

A aplicação poderá considerar também ser adaptativa às condições da rede. As informações de suporte são enviadas num modo “conta-gotas”, se a rede está com alta capacidade (conectividade forte), se a rede está com conectividade fraca, o envio se dará por solicitação do fiscal, quando este necessitar da informação. Se o dispositivo do fiscal tem capacidade de armazenamento, um *prefetching* dos arquivos, com informações de suporte, é disparado e executa em *background*.

Uma opção auxiliar é informar o itinerário do fiscal, previsto para o dia. Com a informação de itinerário e o perfil do fiscal, pode-se usar técnicas de predição e *prefetching* de dados para a entrega das informações necessárias para a aplicação.

Neste caso, a mobilidade não é considerada um problema a ser transposto, mas uma característica natural do ambiente que deve ser explorada para fornecer maior flexibilidade e utilidade ao usuário final. O usuário que se desloca necessita ter acesso a informações úteis para a posição em que se encontra. A seleção de informações úteis deve ser fornecida pelo sistema e não realizada pelo usuário, como o sistema atual prevê.

Salienta-se que esta aplicação está em fase de estudo de viabilidade, enquanto que a versão nômade está em fase de desenvolvimento pelo CPD-UFSM.

Futuros experimentos desta e de outras aplicações permitirão um entendimento melhor de como a sensibilidade ao contexto pode ser usada e como esta interfere na modelagem da aplicação.

7 TRABALHOS RELACIONADOS

A área de Computação *Pervasiva* é muito recente. Devido à dificuldade em encontrar referências relativas a ela no início do projeto ISAM, em 2000 e 2001, usou-se como base para o levantamento bibliográfico trabalhos que abordam o conceito de mobilidade de diferentes formas. Uma abordagem próxima a da arquitetura ISAM, e mais especificamente de ISAMadapt, ainda não foi encontrada na literatura. Desta forma, neste Capítulo serão discutidos os principais trabalhos que influenciaram as soluções adotadas no projeto ISAMadapt. Destacam-se os temas: adaptação, ambientes *pervasivos*, reconhecimento de contexto e linguagens adaptativas.

7.1 Sistemas Adaptativos para a Computação Móvel

Adaptação é uma estratégia muito adotada em sistemas distribuídos (STEENKISTE, 1999). No ambiente móvel, várias arquiteturas e sistemas para suporte ao comportamento adaptativo das aplicações têm sido propostas, entre estas: Rover (JOSEPH; TAUBER; KAASHOEK, 1996), Coda (SATYANARAYANAN, 1996b), Mobiware (ANGIN, et al., 1998), Timely (BHARGHAVAN; GUPTA, 1997), Odyssey (NOBLE, 2000), MoleNÈ (ANDRÈ; SEGARRA, 2000b). Com foco específico em Computação Pervasiva existem ainda poucos trabalhos, destacando-se: Gaia (ROMAN, 2002) e Aura (GARLAN, SCHMERL, 2002). Estes esforços abordam a mobilidade e suas conseqüências sob diferentes perspectivas, o que dificulta uma análise comparativa mais aprofundada.

Nos trabalhos analisados, pode-se distinguir duas **abordagens da mobilidade**:

a) baseada em mascaramento – tentam mascarar a mobilidade para o usuário no nível de rede ou de sistema, como Most (FRIDAY, 1996), Rover (JOSEPH; TAUBER; KAASHOEK, 1996) e Coda (SATYANARAYANAN, 1996b). Procuram fornecer o mesmo conjunto de serviços de um *desktop*, não requerendo alterações nas aplicações existentes, que podem executar no ambiente móvel mesmo sem terem sido projetadas para ele. Como o mascaramento completo não é possível, as soluções não podem tratar todas as variações. Por exemplo, em Coda, o usuário deve resolver os conflitos de atualizações das réplicas dos arquivos móveis. Como essas plataformas oferecem comunicações baseadas em RPC, relaxam a natureza síncrona das interações, permitindo que as mensagens sejam atrasadas (Most) ou enfileiradas (Rover). Entretanto, o modelo de programação permanece com sua natureza síncrona, pois as plataformas tentam manter a semântica de RPC em face às variações na conectividade da rede. Além disso, a comunicação é orientada a conexões: clientes selecionam serviços a serem usados, ligam-se às suas interfaces e invocam operações nessas interfaces. Outra desvantagem é que esta abordagem não permite projetar aplicações conscientes do contexto;

b) baseada em adaptação – implementam implícita e/ou explicitamente a adaptação, inserida na aplicação. Informações do ambiente são fornecidas para a aplicação que pode se adaptar às novas condições. Permitem implementar adaptações mais relevantes devido a sua integração com a aplicação, porém a maioria das implementações não é flexível nem extensível, uma vez que a adaptação é intrínseca (interna) à aplicação. Em particular, elas somente tratam com recursos monitorados em baixo nível e não se beneficiam da semântica dos dados da aplicação para influenciar a adaptação.

Muitos dos sistemas desenvolvidos para a Computação Móvel enfocam aspectos particulares do problema contido na abordagem adaptativa. Os primeiros projetos (SATYANARAYANAN, 1996b; JOSEPH; TAUBER; KAASHOEK, 1996; TERRY et al., 1999), durante a década de 90, abordaram o tema do acesso aos dados a partir de dispositivos móveis, dando origem aos sistemas de Banco de Dados Móveis atuais. Estes sistemas objetivam a transparência da mobilidade. Um estudo, sobre esse assunto, é encontrado na referência (AUGUSTIN, 2000).

Diferentemente do foco em mascaramento da década passada, atualmente o foco das pesquisas concentra-se em sistemas e arquiteturas que exploram e expõe o conceito de mobilidade/localização e contexto para a produção de softwares móveis que fornecem um comportamento adaptativo. Estes visam atender os requisitos das novas aplicações (AUGUSTIN, 2002b). Dentre as soluções propostas, fez-se uma seleção buscando trabalhos que tem uma relação mais direta com o tema do ISAMadapt: programação da adaptação e consciência do contexto. A análise destes trabalhos é destacada nos próximos itens.

7.2 Programação da Adaptação no Ambiente Móvel

Suporte à adaptação para aplicações móveis tem sido a motivação de muitos projetos que propõem soluções diversas e adotam diferentes cenários de mobilidade (ver Apêndice A). Abordagens incluem desde o uso de estratégias que implementam um comportamento adaptativo da aplicação móvel (NOBLE, 2000; BAGGIO, 1999), passando por suporte à programação de aplicações (DEY, 2000; WELLING; BADRINATH, 1998; BHARGHAVAN; GUPTA, 1997), até a análise e revisão dos paradigmas para torná-los adequados a este ambiente (CAPRA; MASCOLO; EMMERICH, 2003; EFSTRATIOU et al., 2000). Além disso, o suporte à programação pode oferecer abordagens específicas para alguns domínios de aplicações, tais como Odyssey (NOBLE, 2000) e Cadmium (BAGGIO, 1999), ou genéricas, como (ZACHARIADIS; CAPRA; MASCOLO; EMMERICH, 2002 ; ANDRÈ; SEGARRA, 2000a).

7.2.1 Estratégias de Adaptação

Aplicações podem adaptar-se às condições do ambiente em uma variedade de modos. A primeira possibilidade estudada na década de 90 foi relativa **à alteração no modo como os dados são enviados** (AUGUSTIN, 2000). Por exemplo, reduzindo o volume de dados (tamanho do frame de um vídeo, extraindo figuras de um documento, extraindo somente a parte de texto de um *postscript*); ou aplicando compressão (com perda). Esses exemplos devem balancear entre os recursos para computação e os recursos para comunicação. Outra possibilidade é o **uso de métodos específicos do domínio da aplicação**. Por exemplo, quando uma imagem deve ser enviada, pode-se transmitir a imagem renderizada ou os comandos de renderização, dependendo da largura de banda e dos recursos computacionais disponíveis. As computações

distribuídas podem implementar o conceito de adaptação alterando quantos nós elas usam, decidindo executar tarefas local ou remotamente, ou determinando o número ótimo de nós para a computação.

A estratégia mais usada nas aplicações estudadas é implementar adaptação como a **troca de um recurso** ou por outro recurso ou pela qualidade do serviço prestado, dependendo dos objetivos e domínios da aplicação (NOBLE, 2000; BAGGIO, 1999). A adaptação pode ser realizada usando filtragem ou redução – reduzir a quantidade de dados a serem transmitidos, reduzir o número de informações sobre os objetos e/ou eliminar dados menos relevantes (HEUER; LUBINSKI, 1998); transformação – alterar o tipo de dado para reduzir o volume dos dados ou adaptar o formato dos dados para ser mais fácil apresentá-los ao cliente; *caching* – adicionam assincronismo na comunicação, e podem ser implementadas através de estratégias, por exemplo: *caching* sob demanda pode ser usada para minimizar a quantidade de dados transmitidos sobre um link com baixo QoS, enquanto que *prefetching* pode ser utilizando se o link corrente está bom. Essas funções podem ser combinadas. O processo de adaptação é automaticamente disparado pela alteração em um recurso monitorado, como a largura da banda, e/ou o dispositivo *display* em uso (ZENEL; DUCHAMP, 1997).

Outra estratégia comum de adaptação no ambiente móvel é o sistema final **instanciar serviços de *proxies* locais** para tratar com as requisições da aplicação (FOX et al., 1998; ZENNEL; DUCHAMP, 1997; WANG, 1999). Um *proxy* é usualmente definido como uma entidade inserida no fluxo de comunicação em algum lugar entre o cliente e o servidor. O *proxy* pode, então, modificar o fluxo de comunicação (do cliente para o servidor e vice-versa) a fim de adaptar-se à corrente característica de QoS, por exemplo.

Particionamento da aplicação é um conceito onde a funcionalidade da aplicação é dividida ou replicada entre o cliente móvel e a rede fixa baseada em condições de eficiência da rede. A maioria das aplicações móveis correntes são particionadas estaticamente (em tempo de compilação) entre o cliente móvel e o *proxy server*, que é conectado à rede fixa e atua como servidor para o cliente. O particionamento dinâmico permite a divisão em tempo de execução. Rover (JOSEPH; TAUBER; KAASHOEK, 1996) e Kunz (KUNZ; BALCK, 1999) incorporam este conceito. O particionamento pode ser dirigido pela aplicação (JOSEPH; TAUBER; KAASHOEK, 1996) ou ser transparente à aplicação (KUNZ; BALCK, 1999). Para habilitar o particionamento dinâmico, Kunz (1999) estrutura a aplicação em um conjunto de objetos que se comunicam por invocação de métodos. Na criação, cada objeto pode registrar-se no *run-time*, os registros são da forma “execute no móvel se a banda for maior que x” ou “execute na mesma localização que o objeto O”. Uma política *default* trata os objetos sem registro explícito e fornece informações específicas. Por exemplo, objetos que excedem um tamanho limite executam no *proxy server*, outros menores no *host* móvel. O *run-time* monitora os parâmetros relevantes e inicia o processo de migração do objeto, se necessário. A migração é realizada usando a tecnologia de código móvel Java.

Outra estratégia de adaptação usada é obter o **código sob demanda (**FUGGETTA; PICCO, VIGNA, 1998; VIGNA, 1998). Esta é uma forma natural de manter a aplicação em execução com tamanho reduzido e reduzir o tráfego na rede. Outras vantagens no ambiente móvel são (a) reduz o impacto dos recursos limitados do dispositivo móvel, desde que o código não necessita ficar armazenado localmente; (b) o ambiente móvel é caracterizado pela alta escalabilidade (número elevado de clientes móveis), evita instalação de novas versões, já que esta será feita automaticamente; (c) permite minimizar a complexidade e tamanho do sistema e reduzir a dependência de recursos locais nos nodos móveis. A decisão quando/de onde/para onde carregar (e descarregar)

componentes de software depende e requer conhecimento sobre o estado atual dos recursos, como banda, energia e disco. Estratégias podem incluir tecnologias *pull* e/ou *push*. O componente pode não saber a exata localização do recurso remoto que necessita, o que exige possibilidade de pesquisa. Pode ser necessário trazer a ferramenta de software necessária junto ao código que a utiliza. Dado os recursos limitados será necessário um gerenciamento de recursos.

Por outro lado, alguns autores propõem o uso da **execução remota** como uma estratégia de adaptação, a qual objetiva reduzir a comunicação entre nodos móveis e a rede estática. Neste caso, a adaptação é alcançada pela troca do modo como a rede é usada. Sumatra (RANGANATHAN; ACHARYA, SALTZ, 1997), que monitora os recursos da rede para as aplicações móveis, considera a **mobilidade como um mecanismo de adaptação**. A fim de se adaptar às variações das redes, o programa móvel deve ser capaz de decidir quando se mover, o que mover e para onde se mover. Com informações on-line sobre a disponibilidade e qualidade dos recursos, estratégias baseadas em mobilidade podem automaticamente determinar sítios apropriados para localização das estruturas de dados e computações que governam o desempenho da aplicação. Em Sumatra, a comunicação e a migração acontecem sob controle da aplicação (primitiva *go*).

A **reconfiguração da aplicação**, adicionando, removendo ou migrando componentes é a proposta de DACIA (LITIU; PRAKASH, 2000). A aplicação é vista como um grafo dirigido de componentes conectados, que pode alterar sua estrutura em tempo de execução, usando diferentes componentes ou conectando-os de modos diferentes. Em DACIA, um componente é um PROC (*processing and routing component*), conectado a outro seguindo regras e restrições, e com capacidade de ser relocado. DACIA fornece (i) uma máquina (*engine*) que executa em cada nodo fixo ou móvel, e que administra as PROCs, gerencia conexões e relocação (reconfiguração dinâmica); (ii) um monitor que obtém dados de desempenho da aplicação e implementa relocação específica da aplicação e políticas de reconfiguração. Aplicações podem ficar estacionadas (*parking*) enquanto o nodo está desconectado ou ocioso. Estas continuam operacionais porém com algumas limitações, e interagem em nome do usuário. DACIA fornece primitivas para conexão (*addProc, removeProc, connectProc, disconnectProc*), migração (*moveProc*) e comunicação via troca de mensagens (*output*, *input*). *Engine* e PROCs são de propósito-geral e podem ser reusadas para construir outras aplicações; no entanto, o monitoramento é específico da aplicação. PROC's são implementadas em Java, e executam em PC's e PDA's com suporte a *personalJava*.

7.2.1.1 Análise Comparativa com ISAMadapt

No ISAM, procura-se ofertar suporte para o uso de variadas técnicas de adaptação, com vistas ao projeto de aplicações *standalone*. Código sob demanda e alocação automática baseada no ambiente corrente são estratégias inerentes à arquitetura; migração, replicação, reescalonamento, e outras técnicas estão inseridas como construções para expressar um comportamento adaptativo na linguagem ISAMadapt. Desta forma, a aplicação contém código orientado à adaptação, diferenciando-se dessas abordagens que são específicas em sua maioria.

7.2.2 Suporte à Programação da Adaptação

Do ponto de vista de desenvolvimento e manutenção, existe a vantagem de organizar a aplicação. Pode-se adotar duas abordagens para organizar a aplicação: construir a aplicação inteira tendo a adaptabilidade em mente (embutir a adaptação dentro da aplicação) ou isolar a adaptação da lógica da aplicação. Essas duas abordagens não são

necessariamente contraditórias e podem coexistir em uma aplicação. A primeira opção é abordada em Rover (JOSEPH; TAUBER; KAASHOEK, 1996), Odyssey (NOBLE, 2000) e Cadmium (BAGGIO, 1999). O segundo enfoque é adotado em sistemas reflexivos (EFSTRATIOU et al., 2000) e também em MoLèNE (ANDRÈ; LE MOÜEL, 2000), onde objetos de contexto encapsulam a parte adaptativa da aplicação, o comportamento padrão da aplicação permanece independente.

Tornar uma aplicação adaptativa pode requerer uma profunda alteração na sua estrutura. Para amenizar essa exigência, pode-se embutir o tratamento da adaptação em sistemas de suporte, que apresentam diferentes arquiteturas. O suporte do sistema pode ser oferecido na forma de *run-time* com APIs (JOSEPH; TAUBER; KAASHOEK, 1996; BHARGHAVAN; GUPTA, 1997; BAGGIO, 1999; NOBLE, 2000), *toolkits* com *frameworks* (WELLING; BADRINATH, 1998; ANDRÈ; LE MOÜEL, 2000; DEY, 2001; DEY, 2000), ou *middlewares* (EFSTRATIOU et al., 2000; PICCO; MURPHY; ROMAN, 2001; ZACHARIADIS; CAPRA; MASCOLO; EMMERICH, 2002).

7.2.2.1 APIs

Em geral, os primeiros sistemas móveis ofereciam uma interface de acesso às rotinas de bibliotecas/*run-time*, como em Odyssey (NOBLE, 2000), Cadmium (BAGGIO, 1999), Timely (BHARGHAVAN; GUPTA, 1997). Em Cadmium, a adaptação está associada a um conjunto de aspectos (instanciação, replicação, controle da replicação, controle do acesso ou consistência) governados por estratégias - algoritmos ligados dinamicamente (por exemplo, *prefetching* é uma estratégia para replicação) que se adaptam às alterações do ambiente. Uma visão diferente é dada em Odyssey, que também focaliza o acesso a dados, porém a adaptação é definida através do conceito de fidelidade – grau de semelhança com a cópia no servidor. Fidelidade pode ter muitas dimensões, como consistência (relaxada quando a conectividade é fraca) ou qualidade da imagem e *frames* por segundo, para dados de vídeo. A seleção da fidelidade é uma divisão de responsabilidade entre o servidor e a aplicação em face à alteração no(s) recurso(s) monitorado(s). Odyssey e Cadmium fornecem políticas específicas de acesso aos dados e um subsistema de detecção/notificação das alterações. À aplicação cabe informar quais alterações são significativas para ela, e a política a ser usada no contexto corrente. Como o subsistema de detecção está embutido na adaptação, a funcionalidade global torna-se rígida, é difícil definir novos recursos e variações, tornando esta abordagem não extensível. Além disso, grande parte do trabalho fica para o programador da aplicação.

7.2.2.2 Frameworks

Para simplificar a tarefa de programação pode-se usar ***frameworks*** para capturar o fluxo de controle do processo de adaptação; a aplicação somente preenche os componentes específicos dela. Alguns exemplos desta solução são descritos em (BOLLIGER; GROSS, 1998; WELLING; BASRINATH, 1998; ANDRÈ; SEGARRA, 2000). Bolliger (1998) propõe um *framework* no qual a adaptação está relacionada à qualidade percebida pelo usuário. Somente a aplicação (projetista) sabe o que qualidade significa. Por exemplo, a aplicação pode sacrificar algum grau de qualidade para obter um tempo de resposta melhor, ou esperar um pouco mais para obter um resultado melhor. Neste *framework*, a adaptação é relativa a dados, num sistema de laço fechado contendo três fases: monitoramento e reação; preparação e transmissão. Quando a adaptação é necessária, a primeira fase decide que objetos ajustar e quais as transformações a aplicar; a segunda, realiza a transformação; e a terceira, envia os objetos transformados para o cliente.

Welling (1998) apresenta um *framework* onde o ambiente é monitorado e a aplicação é notificada através de **eventos assíncronos**. Um evento é um objeto de alto nível que encapsula informações relativas às alterações no ambiente que este representa. Eventos podem ser: o *host* foi movido, ocorreu um *hand-off*, um cartão PCMCIA foi inserido/removido, a qualidade do sinal foi deteriorada, a bateria está fraca, uma nova rede foi detectada, uma banda melhor está disponível, a banda deteriorou, etc; e é responsabilidade do programador da aplicação tratar esses eventos definindo os tratadores de eventos.

Nesta mesma linha, a abordagem do sistema MoLèNE (ANDRÈ; SEGARRA, 2000) oferece um conjunto de serviços genéricos, modelados em uma arquitetura dividida em um *toolkit* e uma camada *framework*. O primeiro fornece gerenciamento dos dispositivos relacionados aos serviços; o segundo fornece serviços correspondendo a uma estrutura de classes com métodos abstratos que deverão ser definidos pela aplicação. Assim, o comportamento dos serviços projetados pode ser modificado dependendo das necessidades da aplicação. No modelo do sistema, um processo intermediário (*middleprocess*) fornece a funcionalidade para as aplicações executando no dispositivo móvel e gerencia as questões de mobilidade. Para realizar a adaptação, o *framework* é estruturado em três tipos de objetos: serviço, micro-serviços e implementação. A adaptação é realizada alterando-se a implementação e/ou localização dos micro-serviços, ou alterando-se a estrutura do serviço, adicionando/removendo seus objetos micro-serviços constituintes. As aplicações cobertas por MolèNE são os sistemas de gerenciamento de dados no ambiente sem fio. O sistema gerencia a heterogeneidade dos sistemas de armazenamento (interface, unidades de informação,...) e fornece serviços como *Networking Service, Caching Service, Consistence Service* e *DataAccess Service*.

Em Timely (*The Illinois Mobile Environments Laboratory*) (BHARGHAVAN; GUPTA, 1997) a aplicação distribuída é modelada em três entidades: o cliente consciente de QoS, o servidor, e *stubs* da aplicação conscientes de QoS que executam no *home*. O modelo de adaptação consiste de três camadas: a camada de baixo nível trata dos recursos de rede (capacidade do canal, *buffer*...) e de computação (bateria, memória, disco, CPU,...); a camada intermediária gerencia os diferentes recursos e fornece o suporte adaptativo para a camada de aplicação. A característica chave do *framework* é que a camada de **gerenciamento de recurso** fornece os mecanismos para gerenciar, monitorar e reagir a alterações na disponibilidade do recurso, enquanto que a camada de aplicação fornece a política para tratar dinamicamente essas trocas. Para a implementação do comportamento adaptativo da aplicação, dois conceitos são introduzidos: classes de QoS e blocos de adaptação. Uma classe de QoS é definida especificando os limites máximo e mínimo para os recursos (parâmetros de QoS previstos: razão de dados, atraso, probabilidade de *handoff* e custo por byte). A aplicação divide sua execução em **blocos de adaptação** – conjunto de seqüências alternativas de execução, cada uma associada com uma classe de QoS. A aplicação negocia os recursos no início (*start*) de cada bloco de adaptação, e espera que esta seja mantida até o final da execução do bloco. Porém, se for notificada de uma violação na qualidade, o sistema pode renegociar no meio do bloco de adaptação.

7.2.2.3 Middlewares

Os *middlewares* permitem a portabilidade das aplicações, através das plataformas heterogêneas das redes e dos sistemas operacionais, pois o *middleware* pode encapsular as peculiaridades das plataformas. Essas tecnologias tem sido projetadas com sucesso para sistemas distribuídos estacionários, executando sob redes fixas (EMMERICH,

2000). Porém, a simples adoção da tecnologia atual de *middlewares* não é adequada ao ambiente móvel. Primeiro, as primitivas de interação, tais como transações distribuídas, requisições a objetos ou chamada remota de procedimento, assumem uma alta largura de banda, e conexão permanente e disponível. Isso contrasta com as características inerentes ao ambiente móvel. Segundo, *middlewares* orientados a objetos, como CORBA e Java-RMI, suportam principalmente comunicação ponto-a-ponto síncrona, a qual requer que o cliente solicite um serviço e o servidor o atenda, ambos executando simultaneamente. Novamente, contrasta com o ambiente móvel que requer anônima e assíncrona comunicação. Terceiro, ambientes distribuídos assumem que o ambiente de execução é estacionário, com confiável e alta largura de banda, localização fixa de cada nodo, e serviços bem conhecidos. Esta visão contrasta com o cenário altamente dinâmico proposto pela Computação *Pervasiva*, onde a localização dos nodos altera-se no tempo, e novos serviços podem ser descobertos dinamicamente enquanto este se move. Finalmente, a carga computacional para execução de *middlewares*, como CORBA, é alta para ser carregada e executada em nodos móveis.

O corrente estado-da-arte dessas plataformas fornece paradigmas de programação orientadas a conexão, refletindo sua origem nas redes fixas. Porém, as aplicações móveis requerem paradigmas de programação assíncrono, não orientado a conexões, com suporte generalizado para controle e monitoramento de recursos e serviços.

Em face dessas dificuldades, novos *middlewares* estão sendo propostos. Algumas soluções enfatizam alterações nas plataformas existentes, como o projeto DOLMEN (www.fub.it/dolmen) e ALICE (www.dsg.cs.tcd.ie/research/alice) para alterações no padrão CORBA, ou propõe novas plataformas para suportar configuração dinâmica, operação desconectada e adaptabilidade. Sistemas que abordam um aspecto específico também começam a serem disponibilizados: mucode (PICCO, 1998) para mobilidade de código baseado em três conceitos: *classSpace, classLoader, Server*, que permitem implementar uma base de código *pervasiva*; descoberta de recursos e serviços (RAKOTONIRAINY; GROVES, 2002).

Por outro lado, Roman (ROMAN; PICCO; MURPHY, 2000) propõe o desenvolvimento de novos *middlewares*, desenvolvidos desde seu início com mobilidade física em mente. Essas novas plataformas são agrupadas por ele em três categorias: (i) reflexiva, que tentam construir *middlewares* leves e reconfiguráveis (BLAIR, 1998; CAPRA; MASCOLO; EMMERICH, 2003); (ii) espaço de tuplas, que tentam resolver o problema da desconexão e comunicação assíncrona de forma natural (PICCO; MURPHY; ROMAN, 2001); (iii) conscientes do contexto, que tentam fornecer informações do contexto para as aplicações.

A proposta de Capra (CAPRA; 2001) é a associação de **reflexão**, técnicas de código móvel e metadados para a construção de uma plataforma de *middleware* para a Computação Móvel. Os metadados permitem a separação de conceitos, distinguem o que o *middleware* faz de como faz. O papel da reflexão nos *middlewares* é trazer maior abertura e flexibilidade à plataforma, uma vez que um dos problemas para sua adoção na Computação Móvel é que a plataforma *middleware* esconde os detalhes de implementação e toma decisões em nome da aplicação. Um sistema reflexivo permite modificar a plataforma através da inspeção e da adaptação (BLAIR, 1998a). Com a inspeção, o comportamento interno do sistema é exposto, e pode-se inserir um comportamento adicional para monitorar a implementação do *middleware*. Desta forma, poder-se-ia oferecer informações de contexto para a aplicação. Através da adaptação, o comportamento interno do *middleware* pode ser alterado dinamicamente. Um exemplo de adaptação é reduzir a carga computacional do *middleware* instalando no nodo móvel

somente um núcleo de funcionalidades, e fazer carga de código sob demanda de acordo com as necessidades da aplicação em execução.

Recentemente, atenção tem começado a focalizar o uso do **modelo de interação assíncrona** para suporte as aplicações móveis adaptativas. Os *middlewares* baseados no conceito de espaço de tuplas (BELLAVISTA; CORRADI; STEFANELLI, 2001; PICCO; MURPHY; ROMAN, 2001; DAVIES; FRIDAY; WADE; BLAIR, 1997) enfatizam o estilo desacoplado e oportunista da comunicação. Desacoplado no sentido que a comunicação acontece em face às desconexões, e oportunista porque explora a conectividade se esta torna-se disponível. Entretanto, a implementação tradicional do espaço de tuplas não atende esses requisitos. Existem questões que devem ser respondidas: como é organizado o espaço de dados globalmente compartilhado entre nodos? Como tornar as tuplas disponíveis e acessíveis de forma altamente distribuída? Um exemplo é Lime (*Linda in a Mobile Environment*) (PICCO; MURPHY; ROMAN, 2001) que oferece um conjunto mínimo de construções para tratar com mobilidade física e lógica. Lime retém a filosofia e objetivos de Linda (GELERNTER, 1985) enquanto adapta-a à mobilidade. O movimento, físico ou lógico, resulta em alterações implícitas do Espaço de Tuplas (ET) acessível aos componentes individuais. O sistema, não o programa de aplicação, é o responsável por gerenciar o movimento e reestruturar o ET associado com as alterações de conectividade. Em Lime, agentes móveis contendo seus espaços de tuplas locais viajam entre host e formam um Espaço de Tuplas Transiente Compartilhado, que incorpora o conceito de mobilidade física e lógica.

Outro aspecto importante é relativo à questão **transparência x consciência**. *Middlewares* são construídos em abordagens que enfatizam a transparência, onde programadores não necessitam conhecer nenhum detalhe sobre o objeto ao qual o serviço é requerido. Enquanto que em sistemas distribuídos estacionários é possível (e desejável) esconder completamente as informações de contexto e detalhes de implementação da aplicação, em ambientes móveis isto se torna mais difícil e inadequado. Para oferecer transparência, os *middlewares* 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 de cada aplicação, o nodo móvel e o ambiente de execução corrente. Em sistemas móveis é essencial que o *middleware* seja adaptativo. Os *middlewares* existentes não provêem suporte a consciência do contexto, pois foram projetados para escondê-la do usuário.

Middlewares para fornecer **consciência de contexto** às aplicações começam a serem propostos. Porém abordam, em geral, somente um ou outro elemento de contexto. Um exemplo de *middleware* que suporta consciência de localização para aplicações é Nexus (FRITSCH; KLINEC; VOLZ, 2000), que objetiva fornecer suporte a um ambiente heterogêneo de comunicação. Diferentemente, Carisma (*Context-Aware Reflective mIddleware System for Mobile Application*) (CAPRA; MASCOLO; EMMERICH, 2003) é um *middleware* que usa reflexão estrutural e comportamental para se reconfigurar e atender as necessidades de cada aplicação - o *middleware* torna-se consciente do contexto. Por exemplo, o serviço de acesso a dados pode ser fornecido por cópia ou link. A aplicação expressa que deseja o acesso por cópia se tem recursos suficiente no dispositivo. Esta customização é baseada no perfil da aplicação, definido pelo programador em XML, que é consultado pelo *middleware* a cada requisição de serviço. O contexto é obtido diretamente do sistema operacional da rede.

Outra direção de pesquisa na área focaliza a construção de middlewares para suporte ao paradigma de **comunicação par-a-par** (P2P: *peer-to-peer computing*). Computação por pares tornou-se popular com os sistemas Napter (www.napster.com), Gnutella

(www.gnutella.org) e Freenet (http://freenet.sourceforge.net). Esse modelo permite implementar aplicações distribuídas onde cada usuário compartilha, com todos os pares conectados, transparentemente, informações armazenadas localmente. A organização é altamente distribuída, sem um servidor central, e em larga escala. Informação 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. Este modelo é particularmente apropriado para **redes *ad-hoc***, onde a comunicação é habilitada em todos os níveis somente por nodos móveis. Nestas redes, as interações nodo-nodo são seu modo de vida, desde que constituem a forma fundamental de comunicação. Similarmente, a habilidade de consultar o espaço global de informação assume um papel importante desde que o conjunto de pares conectados altera-se dinamicamente e qualquer tipo de referência pode tornar-se obsoleto e/ou ficar indisponível rapidamente (CUGOLA; PICCO, 2001a). Cada nodo necessita hospedar suporte *run-time* para gerenciar o roteamento de mensagens, a notificação de eventos e requisições para executar operações locais. Porém, o custo computacional ainda é alto para manter e obter informações globais.

7.2.2.4 Análise Comparativa com ISAMadapt

No ISAM, usa-se uma integração de muitos conceitos presentes nestes trabalhos para oferecer abstrações de mais alto nível, embutidos numa linguagem de programação, cujo ambiente de execução torna transparente para o programador as questões de gerenciamento da mobilidade e da adaptação. API's de programação são fornecidas pelo `ISAMcontextService` e utilizadas pelo tradutor do programa ISAMadapt e pelo EXEHDA. A estratégia de *middleware* é utilizada para implementação do ambiente de execução EXEHDA. Um *framework* está embutido na metodologia de projeto e programação implementada pelo ISAMadapt.

7.2.3 Domínio de Aplicações

Um grande número de aplicações legadas tem pouca adaptação pré-embutida, o que sugere uma preferência para arquiteturas que isolam a adaptabilidade, como as reflexivas. A grande maioria dos sistemas propostos aborda aplicações em domínios específicos, tais como a multimídia em Mobiware (ANGIN et al., 1998), e acesso a dados em Cadmium (BAGGIO, 1999) e Odyssey (NOBLE, 2000). A adaptação ocorre transformando o formato dos dados, considerando a banda disponível na transmissão. Sumatra (RANGANATHAN; ACHARYA; SALTZ, 1997) implementa *adaptalk*, um *chat* cujo servidor central (*chat-board*) é móvel e se localiza próximo aos clientes. As informações de latência fornecidas por um monitor, chamado Komodo, dirigem a localização do servidor. O domínio coberto pelo sistema MoLèNE (ANDRÈ; SEGARRA, 2000) são aplicações distribuídas de dados intensivos, como videoconferência e comércio eletrônico via Internet. Os serviços oferecidos incluem acesso a dados remotos, *caching*, consistência e serviços de rede. Outra área de aplicação são os sistemas colaborativos (multiusuários) como em Limbo (DAVIES; FRIDAY; WADE; BLAIR, 1997), *groupware* e *workflow*, como discutido em (JING; KUFF, 1998). Diferentemente, outros trabalhados focalizam a adaptação em aplicações dependentes de localização (*location-aware applications*) (TSENG; WU; LIAO; CHAO, 2001; COMPUTER, 2001).

7.2.3.1 Análise Comparativa com ISAMadapt

ISAM não está comprometido com um domínio específico de aplicação, pois visa dar suporte no nível de linguagem para aplicações *pervasivas* de propósito-geral que exibam um comportamento consciente do contexto.

7.2.4 Resumo Comparativo ISAMadapt e Sistemas Móveis Adaptativos

A seguir, no Quadro 7.1 faz-se uma comparação entre trabalhos selecionados que abordam adaptação à mobilidade e o ISAMadapt.

	Suporte às Fases da Adaptação			Abordagem do desenvolvimento				Métrica para adaptação	Estratégia de Adaptação
	M	N	R	R	B	L	F		
Cadmium	x	x			x			Condições da rede	Aspectos da replicação, como *prefetching*, são governados por estratégias – algoritmos plugados dinamicamente
Odyssey	x	x	x	x	x			Qualidade dos dados desejada pelo usuário	Alteração da fidelidade dos dados Transformação dos dados (razão frame, cor,...)
Welling	x	x					x	Eventos assíncronos	Não aborda.
Timely	x	x	x		x		x	Parâmetros QoS (banda)	Serviço adaptativo para gerenciamento de recursos
Sumatra-Komodo	x	x				x		Latência	Migração de *threads* Execução remota
MolèNE	x	x	x				x	*Resources monitors* – mede disponibilidade de recursos como memória, bateria, processador, e obtém informações da rede	Localização do *middleprocess* (migração), implementação variável dos microserviços *caching*, *prefetching*, Filtragem de dados desconexão
ISAMadapt	x	x	x	x		x		Elementos de contexto definidos pela aplicação	Código sob demanda, Migração, replicação, Desconexão, Rescalonamento, Definida pelo usuário

Quadro 7.1: ISAMadapt x Sistemas de Suporte às Aplicações Móveis

Legenda: **M**onitoramento e/ou Gerenciamento de Recursos, **N**otificação de alterações no ambiente, **R**eação às alterações; (ii) para desenvolvimento da aplicação: ***R**un-time* (automática), **B**iblioteca de funções com API para programação, construções de **L**inguagem com correspondente *run-time*, ***F**ramework*.

O Quadro 7.2 apresenta um detalhamento da adaptação em cada um dos sistemas.

7.3 Paradigmas de Programação Distribuída

A adaptação deve ser fornecida em termos de configuração, como uma forma de selecionar operações específicas em tempo de inicialização, e reconfiguração, como uma forma de trocar o comportamento em *run-time*. Considerando o ambiente de rede móvel infra-estruturado do ISAM, em geral, os sistemas se apoiam em alguns modelos de programação, não excludentes: cliente/servidor, objetos, reflexão e aspectos.

	Onde ocorre			Base da Adaptação						
	C	S	P	P	M	B	C	Arquitetura do Sistema	Mecanismos de Notificação	Domínio de Aplicação
Cadmium		x		x	x			Middleware de suporte à replicação, e classes que implementam estratégias e monitoração do ambiente	callback	Acesso a dados
Odyssey	x			x	x			Viceroy – monitor; API para aplicação: negociar recursos, rastrear ambiente, alterar políticas (*ody-tsop*)	Assíncrona: *ody-request*(lim.max,lim.min, ref., tratador); *Ody-cancel*, invocada quando não deseja mais notificação	Acesso a dados Multimídia *Network-aware*
Timely	x			x	x	x		Adaptação em 3 camadas: rede, recurso e aplicação; Classes de QoS e Blocos de Adaptação	Retrieve-QoS	*Netware-aware*; adaptação programada de QoS
Sumatra-Komodo	x				x			Komodo- monitor de latência; Abstrações Java: migração de grupo de objetos, Migração de *threads*	Demanda contínua	*Network-aware* adaptalk
MolèNE			x		x			*Adaptability tool*: Serviços e microserviços. Adaptação altera a localização ou implementação dos microservicos, adiciona/remove serviços	*Detection/Notification Tool* notifica o *Global Adaptor* quando ocorre uma das condições booleanas (registradas) sobre os recursos monitorados	Gerenciamento de dados no ambiente sem fio
ISAMadapt	x	x		x		x	x	ISAMcontextService– contexto geral; Linguagem de Programação, middleware EXEHDA	Tuplas de natureza reativa	*Pervasive applications* Propósito geral, Consciente do contexto corrente.

Quadro 7.2: ISAMadapt x Processo de Adaptação nos Sistemas Móveis

Legenda: onde ocorre a adaptação: **C**liente, **S**ervidor, **P**roxy ou agentes de serviços. A adaptação é baseada em **P**olíticas pré-embutidas, chamada de **M**étodos da interface de programação (biblioteca de suporte), execução de **B**locos de Adaptação com multifuncionalidade, **C**onstruções de linguagens.

7.3.1 Cliente-Servidor

O modelo **cliente-servidor** é largamente usado em muitas aplicações distribuídas, porém no ambiente móvel sofre algumas variações (JING; HELAL, 1999).

Tipicamente, existem três abordagens: uso de procuradores (*proxies*), uso de agentes móveis, e o modelo de subscrição.

7.3.1.1 Cliente-procurador-servidor

O procurador (*proxy*) é um processo intermediário entre o cliente móvel e o servidor que pode executar computação e armazenamento em nome do cliente. A vantagem desta abordagem é que permite ao cliente ou ao servidor permanecer inalterado, ou com pouca alteração. A função e localização do procurador podem variar nos sistemas. Em sistemas mais simples, o procurador executa a adaptação no nível de dados (possivelmente implementado no nível de protocolo), em geral, no servidor, transformando (comprime e/ou filtra os dados, onde o grau de compressão/filtragem depende da banda disponível) determinados tipos de dados para a transmissão, como em (FOX et al., 1998). Sistemas mais flexíveis, como Diana (AHMAD et al., 1995) permitem ao procurador migrar para o nodo móvel, podendo atuar como um servidor leve, de forma a manter a continuidade da computação face à desconexão. Quando a conectividade à rede fica mais forte, o procurador pode migrar novamente para a rede fixa. Porém, este sistema foi projetado para dispositivos móveis com poder computacional, como *notebooks*.

7.3.1.2 Cliente-agente-servidor

Agentes móveis parecem apropriados à Computação Móvel. São entidades que contém código e estado interno, dotadas de autonomia e com a capacidade de migrar entre os pontos da rede, dirigidos por itinerários, em busca dos recursos que a aplicação necessita. Os agentes para a Computação Móvel devem tratar dos problemas que advém da desconexão e fraca conectividade, das restrições naturais dos dispositivos móveis, e da descoberta e gerenciamento dos recursos cuja localização é desconhecida a princípio. Existem muitos sistemas de agentes móveis (PHAM; KARMOUCH, 1998), porém poucos abordam as questões da mobilidade física, dentre esses citam-se: D'Agent (D'AGENT Project, 2000), Concordia (CASTILLO et al., 1998), SOMA (BELLAVISTA; CORRADI; STEFANELLI, 2001). Os agentes podem ser implementados como procuradores ou como agentes de serviços específicos (PITOURA, 1998).

7.3.1.3 Modelo de Subscrição

Uma variante do modelo cliente-servidor é o modelo de publicação-subscrição (*publish-subscribe*). 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 (CUGOLLA; PICCO, 2001). Porém, este modelo foi projetado para redes fixas, com alto acoplamento entre cliente e servidor permanentemente disponível. Portanto, não leva em conta os problemas introduzidos pelo meio sem fio, relativos à reconfiguração dinâmica da topologia da rede, às restrições do ambiente e à mobilidade física dos nodos. Os sistemas existentes consideram que (i) *publishers* e *subscribers* são estacionários, e (ii) o padrão de comunicação é fixo. Para abordar esses problemas, o projeto Jedi (CUGOLLA; NITTO, 2001) adota a topologia infra-estruturada da rede móvel e inclui um componente especial, o *dispatcher*, responsável por gerenciar a distribuição das notificações de eventos para todos os componentes inscritos.

Este modelo torna-se um bom candidato à Computação Móvel porquê: (a) um componente pode atuar no sistema sem ter consciência dos outros componentes. O

conhecimento requerido é sobre a estrutura da notificação dos eventos de seu interesse; b) pode-se anexar um componente na arquitetura sem afetar os demais; (c) a comunicação é inerentemente assíncrona.

7.3.2 Objetos Relocáveis

Em Cadmium (BAGGIO, 1999), Rover (JOSEPH; TAUBER; KAASHOEK, 1996) e MaROS (BAYDERE, 1997) é adotado o modelo de **objetos relocáveis,** que podem migrar para determinadas posições da rede. Em Rover, os objetos migram para fins de comunicação local, enquanto que em MaROS, os objetos migram somente para o servidor com o intuito de processamento. Sumatra (RANGANATHAN; ACHARYA; SALTZ, 1997) combina objetos distribuídos e migração de *threads*, permitindo à aplicação flexibilidade de dinamicamente escolher entre mover dados ou mover a computação. Mobiware (ANGIN et al., 1998) é um *toolkit* construído com a tecnologia de **objetos distribuídos,** usando Java e CORBA, que habilita serviços móveis adaptativos à dinamicamente explorar as propriedades de escalabilidade - intrínsica às aplicações de multimídia móveis - em resposta à variação temporal das condições da rede.

7.3.3 Sistemas Reflexivos e Orientado a Aspectos

Para tratar a adaptação, abordagens baseadas em separação de conceitos, como reflexão e orientação a aspectos, começam a ser estudadas. A separação de conceitos visa tornar explícito o que a aplicação faz e o que depende de alterações no ambiente de execução. Porém, esta separação é somente na etapa de codificação. Para a execução da aplicação, é necessária a composição dos conceitos envolvidos. E é na etapa de composição que a aplicação pode ser adaptável às variações do ambiente.

Sistemas reflexivos são aqueles que podem raciocinar sobre si mesmos (BLAIR, 1998). Isto significa que o sistema tem representação de si mesmo através de estruturas disponíveis em *run-time*. A parte do sistema que executa processamento sobre o domínio da aplicação é chamado de nível-base, enquanto que meta-nível é usado para se referir a parte do sistema cujo sujeito da computação é a representação do sistema em si. Essa separação de conceitos pode ser utilizada para desenvolver aplicações que isolam a questão da mobilidade/adaptação. Porém, poucos trabalham a enfocam, e os que o fazem tratam somente a mobilidade lógica.

A idéia central da programação orientada a aspectos (KICZALES et al., 1997) é separar o código que expressa um aspecto – propriedade do sistema que claramente é separável da unidade funcional, como distribuição, gerenciamento de recursos e concorrência – do código que expressa a unidade funcional. Os aspectos são expressos usando uma Linguagem Orientada a Aspectos, enquanto que as unidades funcionais são expressas usando uma Linguagem de Componentes. Desta forma, podem existir diferentes linguagens orientadas a aspectos, uma para cada tipo de aspecto que se deseja tratar. Por exemplo, AspectJ é um ambiente de programação de aspectos desenvolvido pela Xerox PARC (KICZALES et al., 2002). Este inclui Java como a linguagem de componentes, e existe uma linguagem para o aspecto de sincronização (COOL) e outra para o aspecto de invocação remota (RIDL). Portanto, um sistema orientado a aspectos consiste em um programa-base contendo pontos de união (*join-points*) e programas-aspectos. Esses dois tipos de código são unidos por um combinador de aspectos (*aspect weaver*) em pontos especificados, seguindo instruções expressas em uma linguagem de combinação de aspectos (*weaving language*) específica.

Um aspecto deve ser facilmente identificável, autocontido e alterável. No entanto, o problema com a orientação a aspectos é que alguns conceitos, como mobilidade e

adaptação, não são facilmente fatorados em unidades independentes devido ao inter-relacionamento do sistema inteiro, ou parte deste. Além disso, as Linguagens da Orientação a Aspectos atuais dão suporte somente a aspectos estáticos das aplicações. Isto dificulta a expressão da adaptação modelada no ISAM, uma vez que esta é dinâmica e exige a cooperação do sistema para sua execução. Na literatura não foram encontrados trabalhos que tratam mobilidade e orientação a aspectos. A orientação a aspectos está ainda em sua infância, esta é mais uma área aberta de pesquisa que uma tecnologia pronta para uso (CUGOLA; GHEZZI; MONGA, 1999).

Salienta-se que não foram encontradas publicações que abordem a mobilidade física relacionada à reflexão e à orientação a aspectos.

7.3.4 Análise Comparativa: ISAMadapt e Soluções dentro dos Paradigmas de Programação Distribuída

Como o `ISAMPE` é baseado em uma rede móvel infra-estruturada, ISAM usa o modelo cliente-servidor e objetos relocáveis para a modelagem das aplicações. A solução ISAMadapt, embora não tenha sido influenciada diretamente pelos conceitos da orientação a aspectos, retém semelhanças com esta. Buscou-se separação de conceitos para simplificar o desenvolvimento de aplicações móveis adaptativas. O programa-base, que retém a funcionalidade da aplicação, pode ser associado ao programa escrito em ISAMadapt, constituído dos entes da aplicação. Os programas-aspectos correspondem aos adaptadores, enquanto que os métodos e os entes adaptativos correspondem aos pontos de união. As regras de adaptação correspondem às políticas, de forma mais geral, e aos valores dos elementos de contexto associado ao código de adaptação. A consciência de contexto em nossa abordagem é maior e mais flexível que a solução no paradigma de aspectos. Além disso, apresenta outra vantagem, a de que os pontos de união (métodos e entes adaptativos) são elementos de reconfiguração dinâmica e automática, enquanto que na programação por aspectos estes necessitam de uma linguagem específica para isto (inexistente hoje).

O Quadro 7.3 apresenta uma comparação entre as soluções dentro desses paradigmas e as adotadas no ISAMadapt.

7.4 Ambientes para Computação *Pervasiva*

Somente nesse início de década surgiram projetos que visam desenvolver a infra-estrutura de software para o ambiente *pervasivo*, e estes estão em sua fase inicial (como o projeto ISAM). Dois projetos se destacam: Gaia (ROMAN et al., 2002) e Aura (GARLAN; STEENSKISTE; SCHMERL, 2002).

7.4.1 Projeto Gaia

O projeto Gaia (www.cs.uiuc.edu/gaia) visualiza um 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 (ROMAN, 2003).

Características analisadas	Variações do Cliente-Servidor	Objetos Relocáveis	Reflexão e Orientação a Aspectos	**ISAMadapt**
Entidade de mobilidade	Cliente, procurador	Objetos relocáveis	-	Entes
Entidade de adaptação	Cliente ou servidor	Objeto que migra	-	Entes adaptativos, Métodos adaptativos, adaptadores
Entidade de modelagem	Agentes móveis	Objetos ativos e objetos passivos	Objetos, meta-objetos/aspectos	Entes, adaptadores
Modelo de comunicação	Mensagem , Espaço de Tuplas	Mensagem	Mensagem	Espaço de objetos com natureza reativa
Foco	Distribuição, Comunicação	Comunicação co-localizada	Separação de conceitos	Mobilidade lógica e física de natureza global
Sensibilidade ao contexto	Variação no estado dos componentes da rede	Variação no estado dos componentes da rede	-	Contexto genérico, definido pela aplicação
Suporte para dispositivos móveis	Agentes móveis	-	-	Código sob demanda, adaptado ao dispositivo

Quadro 7.3: ISAMadapt x Soluções nos Paradigmas de Programação Distribuída

Gaia é um *middleware* experimental usado para prototipar o gerenciamento de recursos e fornecer uma interface orientada ao usuário (ROMAN et al., 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 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 objetos distribuídos inclui carregar, descarregar, transferir, criar e destruir todos os componentes e aplicações Gaia. Os serviços são relativos 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 Apresentação (slides) (HESS; ROMAN; CAMPBELL, 2002).

7.4.2 Projeto 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. De forma geral, poder-se-ia dizer que o foco das aplicações deste projeto são correspondentes ao Ambiente Virtual do Usuário, definido no ISAM. É um grande projeto que investiga novas arquiteturas para o ambiente *pervasivo*. Seu foco é no usuário, suas tarefas e preferências (GARLAN; STEENSKISTE; SCHMERL, 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:

a) 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;
b) gerente de tarefas que mantém a representação da tarefa e mapeia entre contexto e preferências do usuário;
c) 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 (GARLAN; STEENSKISTE; SCHMERL, 2002). O foco num novo modelo de programação é a inovação do projeto Aura, que está em estágio de proposta.

7.4.3 Análise Comparativa: ISAMadapt e Ambientes para Computação *Pervasiva*

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). ISAMadapt 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 enfatiza mobilidade do usuário, adaptabilidade e consciência a recursos para execução de tarefas pessoais. O ISAM propõe-se uma visão que integra as questões relativas a *Context-aware, Grid and Mobile Computing* num ambiente comum: computação distribuída em larga escala (*Grid Computing*) de aplicações, de diversos domínios, conscientes da mobilidade (*Mobile Computing*) e conscientes do contexto lógico e físico (*Context-aware Computing*) (YAMIN, 2003). 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, que ainda não foi aprofundada no ISAM, mas cujos componentes para implementação estão sendo construídos pelo EXEHDA (instanciação remota, base de dados e código *pervasiva*, perfil do usuário, contexto). Como o projeto Aura ainda não tem resultados publicados, não se pode fazer uma comparação mais detalhada com o ISAM.

Salienta-se que a proposta da arquitetura ISAM para o ambiente *pervasivo*, com o ambiente virtual e a semântica siga-me das aplicações, é anterior ao conhecimento das publicações do projeto Gaia e Aura. O Quadro 7.4 apresenta uma análise comparativa desses projetos.

7.5 Monitoramento de Contexto

Monitorar o ambiente é uma atividade complexa e cara para os sistemas, porém, é um requisito básico para o comportamento adaptativo das aplicações. Em geral, as informações de contexto utilizadas pelas aplicações móveis já projetadas estão relacionadas à rede (banda, latência, nodos disponíveis, etc.), à localização do usuário ou nodo, às atividades executadas pelos usuários e ao tempo (CHEN; KOTZ, 2000), e são fornecidas:

a) no nível de aplicação. Por exemplo, um componente monitora o fluxo de dados e usa informações desse fluxo para derivar as condições da rede ou da aplicação (COUDERC; KERMARREC, 1999; RANGANATHAN; ACHARYA; SALTZ, 1997; ARAUJO, 2002);
b) no nível de rede. Alguma entidade da rede fornece informação sobre as condições da rede para os emissores, como o protocolo SNMP (*Simple Network Management Protocol*) (DEWITT et al., 1997);
c) no nível de sistema de suporte, que mede alguns parâmetros do ambiente, e notifica a aplicação.

A primeira (a) tem a vantagem de não necessitar suporte adicional, porém é mais complexa de ser implementada. Como é baseado em experiência, severamente restringe o espaço de informação que é explorado; fornece informações ao longo do tempo, mas somente enquanto a aplicação está ativa; e não é útil para decisões de *start-up*. Algumas formas e dimensões de adaptação não podem ser satisfeitas sem medidas externas. Por exemplo, migração dinâmica para um novo conjunto de nós, tanto quanto adição de nós durante a execução, requerem informações externas, pois as informações internas são limitadas aos nós executando correntemente.

Aspectos analisados	Projeto Gaia	Projeto Aura	**Projeto ISAM**
Foco	Espaços de objetos ativos	"aura" de informações pessoais, execução de tarefas pessoais	Integração context, grid e mobile computing para fornecer ambiente de computação pervasiva
Infra-estrutura fornecida	Sistema operacional GaiaOS, e framework para Espaços Ativos	Middleware de gerenciamento de atividades pessoais	Ambiente de Desenvol-vimento com uma Ling. Prog., Middleware de suporte à execução Serviços de reconhe-cimento de contexto
Suporte aos dispositivos móveis	-	Gerenciamento externo controla os dispositivos	Código leve, sob demanda e adaptado ao dispositivo
Suporte à mobilidade de componentes de software	Operação de transfe-rência do Kernel, atua no Espaço Ativo	-	Mobilidade global, lógica e física (dispositivo e usuário)
Suporte à adaptação de componentes de software	Framework para Espaços Ativos	Adaptação é implícita na infra-estrutura	Adaptação negociada com o ambiente de execução, gerenciada pela ISAMadaptEngine, Adaptação explícita através da definição de Comportamentos alternativos, Políticas e comandos de adaptação.
Serviços fornecidos	Repositório de informações, Gerenciamento de recursos, Obtenção de contexto dentro do Espaço Ativo, Gerenciamento de Objetos Distribuídos	Observação de contexto pessoal, Gerenciamento de tarefas pessoais, Gerenciamento do ambiente	Armazenamento pervasivo – ISAMbda, Gerenciamento da adaptação – ISAMadaptEngine, Gerenciamento do ambiente (mobilidade, comunicação,...) – EXEHDA, Reconhecimento do contexto – ISAMcontextService, Ambiente virtual do usuário - AVU
Elementos de contexto considerados	Presença de objetos e pessoas, Condições ambientais, tais como temperatura. Espaços Ativos restritos ao ambiente usado por professores	Relativo ao usuário: tarefas e preferências	Contextos nativos, Contextos definidos pela aplicação (natureza genérica do contexto)
Plataforma de software-base	GaiaOS, CORBA	Java	Java
Suporte ao desenvolvimento de software	framework	-	ISAMadapt IDE com Linguagem de programação
Tipos de aplicações-alvo	Pré-existentes	Aplicações pessoais	Aplicações móveis sensíveis ao contexto de forma genérica
Estágio do projeto	experimental	Em proposta	Em construção

Quadro 7.4: ISAM x Projetos para Ambientes *Pervasivos*

A alternativa (b) permite utilizar protocolos ou sensores disponíveis que fornecem o status do recurso. É mais fácil de usar, mas requer suporte da rede ou sensores, nem sempre disponíveis. A informação fornecida pelo suporte da rede pode ser contínua – baseada em uma propriedade da rede que deve ser interpretada no espaço da aplicação (acoplamento indireto) – ou a aplicação pode ser notificada (acoplamento direto) quando um evento específico acontece (por exemplo, chaveamento de uma rede para outra). Tal notificação deve ser na forma de *callbacks* ou tratadores de eventos específicos. Uma importante desvantagem é a necessidade de a aplicação tratar os dados obtidos, em formatos específicos do protocolo ou do sensor utilizado.

Cada camada da rede pode fornecer informações para a aplicação, entretanto quanto mais baixa a camada mais exata a informação, porém mais difícil de tratar no nível de aplicação. Esta dificuldade conduz à necessidade de uma interface (API) unificada que emprega diferentes mecanismos para obter informações, e as fornece à aplicação, conforme esta as requisita. A opção (c) é adotada no `ISAMcontextService` pela flexibilidade e por permitir um gerenciamento do sistema de suporte.

As aplicações podem usar informação de contexto para vários propósitos. Os principais são: (i) contexto como informação de entrada adicional, por exemplo: uso em algoritmos de filtragem; (ii) contexto como informação que modifica a entrada, por exemplo: uso em aplicações sensíveis à localização; (iii) contexto como informação de *feedback* para o sistema, por exemplo: uso em aplicações baseadas no perfil do usuário; (iv) contexto como gatilho de ações. No ISAM, o contexto é usado principalmente com o propósito (iv), embora projetistas de aplicações possam utilizar as informações com os demais propósitos.

7.5.1 Obter Dados do Ambiente

A noção de contexto é sempre relativa, e usada de maneiras diversas. Contexto pode se referir a fatores humanos, relativos ao usuário, ao ambiente social ou às tarefas que este executa; e fatores físicos, relativos à localização, à infra-estrutura e às condições do ambiente. Em geral, o suporte fornecido à aplicação é em termos de detecção e notificação das alterações ambientais, baseadas em dados sensorados, e deixando a cargo da aplicação a tarefa de definir o que fazer com essas informações (CORDEC; KERMARREC, 1999; MILLER; STEENKISTE, 2000; DEWITT et al., 1997).

Em geral os sistemas monitoram somente determinados elementos de contexto, conforme a necessidade da aplicação-alvo. Por exemplo, `Komodo` monitora latência para a aplicação `adapttalk`, escrita utilizando Sumatra (RANGANATHAN; ACHARYA; SALTZ, 1997); localização do usuário sobre o tempo é usada por muitas aplicações: `Cyberguide` (ABOWD et al., 1997), `Fieldwork` (PASCOE; MORSE; RYAN, 1998); atividade corrente é usada para o `Conference Assistant` (DEY et al., 1999) e `Office Assistant` (YAN; SELKER, 2000).

Diferentemente, Dey (DEY, 2000) forneceu um *framework*, chamado *Context Toolkit*, que influenciou o projeto do `ISAMcontextService` e tem influenciado o surgimento de outros trabalhos: *EgoSpaces* (JULIEN; ROMAN, 2002) e *Context Fabric* (HONG, 2001).

7.5.2 Análise Comparativa: ISAMcontextService e Monitoramento de Contexto

Infra-estrutura de suporte às aplicações conscientes de contexto tem emergido nestes últimos anos. Os sistemas encontrados estão em fase de projeto e/ou implementação. Estas propostas apresentam semelhanças e diferenças com o `ISAMcontextService`.

Todos usam Java como a plataforma de desenvolvimento, e apresentam um modelo informal de contexto. EgoSpaces não diz como os dados de contexto são obtidos, seu foco é como montar uma visão particular dos dados para a aplicação. Usa o conceito de tuplas reativas, como ISAMcontextService, e considera que componentes irão produzir as tuplas de contexto. Context Toolkit e Context Fabric inspiraram-se na funcionalidade das interfaces de usuário, enquanto que ISAMcontextService baseia-se no modelo cliente-servidor. Alguns dos serviços fornecidos pelo ISAMcontextService estão presentes em Context Fabric como abstrações principais: especificação, consulta e notificação por eventos. Context Fabric focaliza na descoberta automática dos sensores baseado na especificação do programador utilizando uma linguagem específica, enquanto ISAMContextService fornece uma ferramenta de navegação na base de componentes para o programador selecionar os elementos que necessita, configurar ou incluir um novo. A linguagem de especificação proposta em Context Fabric é uma variação de XML. ISAMcontextService utiliza XML como formato das informações sobre os elementos de contexto, fornecidas pelo programador usando a interface do ambiente de programação ISAMadapt. Como ISAMcontextService, Context Fabric assume que existem muitos sensores disponíveis e focaliza nas questões relativas ao nível de aplicação, fornecendo abstrações de alto nível aplicáveis à programação de aplicações. ISAMcontextService e Context Fabric separam a especificação do contexto do seu processamento, enquanto que Context Toolkit incorpora-os na aplicação.

O foco do Context Toolkit é numa metodologia de programação para aplicações conscientes de contexto. Grande parte do trabalho fica a cargo do programador que conta com o *framework* para orientá-lo. No ISAMadapt há um suporte mais a visão declarativa da programação, na qual o projetista descreve o resultado desejado do contexto, e usa a infra-estrutura para determinar como isto é alcançado. Por fim, ressalta-se que as aplicações alvos de ambos são as conscientes do contexto, porém com visões diferentes. No ISAM o contexto é o elemento que permite tornar a aplicação *pervasiva* executável nos diferentes ambientes em que se encontra devido ao movimento do usuário, enquanto que no Context Toolkit, o foco é em aplicações interativas que detectam a presença de pessoas ou objetos num lugar e reagem a esta presença. Neste sentido, a solução ISAM é mais genérica, porque permite também a programação de aplicações com o comportamento previsto no Context Toolkit. Além disso, ISAM fornece uma linguagem de mais alto nível para a programação e um ambiente de execução, diferindo da abordagem de *framework* fornecido pelo Context Toolkit. O Quadro 7.5 esboça uma comparação entre a solução ISAMcontextService e esses projetos.

7.6 Linguagens de Programação para Adaptação

Do ponto de vista de programação, os sistemas de suporte à Computação Móvel implementam abstrações como classes em um modelo orientado a objetos, ou rotinas disponibilizadas na API de programação. A abordagem de expressar o comportamento adaptativo via construção de linguagem é inovadora do projeto ISAM. Não foi encontrada nenhuma linguagem adaptativa mais próxima aos nossos objetivos. Desta forma, este item analisa algumas propostas de construções de linguagens que contribuíram com as idéias contidas no ISAM.

aspecto	Context Toolkit	EgoSpaces	Context Fabric	**ISAMcontextService**
Abordagem	framework	Middleware	middleware	Serviços fornecidos pelo middleware
Modelagem do reconhecimento de contexto	Baseado em context widget (semelhante a componentes de interfaces gráficas)	Baseado em agentes (visão particular), local ao nodo/recurso	Baseado em eventos	Baseado no modelo cliente-servidor
Elementos de contexto	Pessoas, lugares e objetos	Topologia da rede, nodos da rede, relativo aos agentes	Não aborda. Considera que outros componentes fornecem a informação de contexto	Elementos lógicos (relativo às aplicações e serviços), físicos (relativo aos recursos, espaço e tempo), pessoal e social (relativos aos usuários)
Representação da informação de contexto	Atributo do context widget	tupla	evento	Tupla (internamente) Informação de alto nível conforme modelada pela aplicação
Serviços fornecidos	Subscrição no context widget Notificação por callback Inspeção de dados sensorados Consulta Armazenamento Refinamento de dados sensorados	Acesso às tuplas com informação de contexto Especificação da visão de contexto	Consulta Notificação de eventos Fornecimento de sensores	Subscrição Notificação via tupla de natureza reativa Consultas Tratamento dos dados sensorados: agregação, tradução para alto nível Predição Descoberta Registro de novos elementos de contexto
Acesso aos serviços	Codificada pelo programador que deve conhecer o nome do widget e a porta que ele escuta	Operações sobre tuplas: Escrita (out) Leitura (rd, in)	Subscrição para receber eventos (análogo a uma interface gráfica)	Realizada pela ISAMadaptEngine que subscreve o ente no ISAMcontextService, e recebe notificação deste
Organização da aplicação	Objetos	Agentes móveis que se comunicam quando fisicamente conectados		Entes (objetos ativos) adaptativos
Declaração do contexto requerido	Codificados pelo programador usando métodos do framework	Agente declara suas necessidades no sistema de manutenção de contexto	Linguagem para especificação de contexto	Comandos do ISAMadapt Definição detalhada do contexto, usando o *context menu* da ISAMadapt IDE
Observações	Tese de doutorado defendida em 2000	Tese de doutorado (em construção)	Proposta de tese (Berkeley)	Projeto financiado (em construção)

Quadro 7.5: `ISAMcontextService` x Projetos de Monitoramento de Contexto

Em geral, as abordagens para introduzir código orientado a adaptação podem assumir três abordagens: imperativa, reflexiva, diretiva. Considerando o exemplo de **alocação** como estratégia de adaptação, tem-se:

a) Imperativa – o código da adaptação assume a forma de instruções nos procedimentos da aplicação. Por exemplo, o comando *move X to locate Y* de Emerald (JUL et al., 1988)[11], um dos primeiros sistemas a introduzir essas construções, especifica ao compilador que o objeto X será movido para o nó onde o objeto Y está localizado; diferentemente, o comando *visit* especifica a migração e, após o término do processamento, o retorno ao nó origem. Outro conceito de Emerald é permitir que um objeto fique imóvel no nó com o comando *fixing*. Além disso, a anotação *attach* na definição de variáveis instâncias permite que objetos formem grupos para fins de migração. Em Emerald toda a alocação é de responsabilidade do programador e não existe nenhuma política de alocação no nível de sistema;
b) Reflexiva – o código da alocação é confinado em meta-objetos associados aos objetos da aplicação. Quando programando o meta-nível, o usuário tem acesso as informações exploradas por ferramentas de alocação automatizadas, tais como o estado dos recursos de hardware, a carga corrente dos nós, a localização dos objetos no sistema. Assim, pode diretamente controlar a localização dos objetos do nível-base pelo comando de suas migrações. No sistema AL/1D (OKAMURA; ISHIKAWA, 1994) os objetos do meta-nível podem armazenar, como variáveis instâncias, informações de carga do estado dos nós do sistema e a localização de outros objetos no sistema. Operações nos objetos do meta-nível, engatilhadas por eventos ocorridos no nível-base, podem acessar estas variáveis e emitir o mecanismo de alocação para implementar políticas específicas. Por exemplo, pode ser definido um método no meta-objeto para reagir a qualquer mensagem enviada e fornecer a migração do emissor para junto do recebedor, sempre que o recebedor for o objeto *X*;
c) Diretiva – o código da alocação assume a forma de diretivas de alto nível que dinamicamente influenciam a alocação dos objetos da aplicação. *Parallel Objects* (CORRADI; LEONARDI, 1997), baseado no modelo de objetos ativos (AUGUSTIN, 1994), trata o problema de alocação integrando a política de alocação no nível de sistema com um conjunto de diretivas de alto nível chamadas ACL (*Abstract Configuration Language*) para especificar as necessidades peculiares dos objetos da aplicação. Exemplos de diretivas: *distributed (#nodes)*: permite aos objetos distribuir seus componentes em muitos nós; *migratable ()*: permite ao objeto migrar; *close_to (Obj)*: especifica a necessidade de um objeto ser alocado próximo ao objeto *Obj*. As diretivas ACL são abstratas e portáveis, pois expressam conceitos lógicos mais do que físicos, forçam a separação entre definição da classe (algoritmo) e necessidades da alocação.

ISAM adota uma integração dessas abordagens: abordagem imperativa (comandos inseridos na linguagem) associada com a diretiva (políticas de adaptação). A abordagem reflexiva - observar e reagir ao ambiente - está presente (em *background*) na consciência ao contexto, subjacente a toda a proposta.

7.6.1 Abordagem da Adaptação nas Linguagens de Programação

Como o exemplo anterior, em geral, linguagens abordam adaptação específica a um aspecto. A linguagem Klaim (*Kernel Language for Agents Interactions and Mobility*)

[11] Considera-se que Emerald usa alocação de processador como estratégia de adaptação ao ambiente.

(BETTINI; LORETTI; PUGLIESE, 2001) oferece construções para tratar o comportamento dinâmico da **conectividade** dos nós (mobilidade física) no nível linguístico. Para tal, introduz uma nova categoria de processo – *NodeCoordinator* – que executa operações para estabelecer conexões (*enter*), aceitar conexões (*accept*) e remover conexões (*exit*). Klaim é inspirada em Linda (GELENTERN, 1988), estendida para tratar com múltiplos espaços de tuplas distribuídos (ET).

Segundo Corradi (2001), o desenvolvimento de aplicações móveis pode se beneficiar, em termos de flexibilidade e dinamicidade, com a adoção de um paradigma de programação que assume a separação explícita da especificação da mobilidade do código da aplicação, onde programadores especificam o comportamento da mobilidade em termos de políticas declarativas de alto nível desacopladas da lógica da aplicação. Desta forma, modificação dinâmica requer somente a troca da política, o controle e a execução da migração são delegados para o suporte à política, transparentemente para o código da aplicação. Uma **política** é um conjunto de regras que governam o comportamento da mobilidade no sistema, especificam quem deve migrar, quando e para onde. O modelo de execução determina como mapear as especificações em implementações das políticas em baixo nível. Sua proposta adota políticas engatilhadas por eventos (*event-triggered policies*) que indicam as ações de **migração** que devem ser executadas quando um evento ocorre. Esta requer um suporte básico para serviços de registro, detecção e notificação de eventos. A linguagem de definição da política Ponder permite expressar os elementos básicos envolvidos na mobilidade: entidade que engatilha a política/ação (*on*); entidade que executa a política (*subject*); entidade que sofre a ação (*target*); ação a realizar (*do*); e evento que dispara a ação (*when*). Por exemplo,

```
Inst oblig P1 {
   on CPU(load, 90);
   subject s = System;
   target t = agent Buyer;
   do t.go(newNode.toString, run())
     when  newNode.isReachable()  -- ação condicionada
}
```

Sumatra (RANGANATHAN; ACHARYA; SALTZ, 1997), uma extensão da biblioteca de classes e do interpretador Java, objetiva investigar programas móveis conscientes da rede (*network-aware*), isto é, programas que podem usar a **mobilidade** como uma ferramenta para se ajustar às variações nas características da rede. As decisões de quando, onde e o que mover são deixadas para a aplicação. Sumatra possui métodos para invocação remota, migração de *threads*, migração de grupos de objetos, e monitoramento de recursos. Para diferentes aplicações, diferentes restrições de recursos governam a decisão de migrar – por exemplo, latência da rede, largura de banda disponível, ciclos de CPU, carga do servidor, etc. Diferente de outras abordagens, Sumatra propõe um único monitor para todos esses recursos, considerando que desta forma é facilitada a implementação de aplicações que necessitam monitorar vários recursos, e que reduz os requisitos de comunicação do monitoramento distribuído.

7.6.2 Análise Comparativa: ISAMadapt e Linguagens para Adaptação

Uma linguagem adequada ao ambiente *pervasivo* que aborde mobilidade e adaptação ao contexto não foi encontrada. Isto impossibilita uma comparação direta entre ISAMadapt e as demais. As linguagens mais próximas abordam expressividade relativa a mobilidade física/conectividade (Klaim), migração baseada no estado da rede (Sumatra), e políticas de mobilidade (Ponder). As abstrações propostas são associadas a uma outra linguagem, como o caso de Ponder ligada a uma linguagem de agentes

móveis chamada SOMA; ou são extensões de linguagens pré-existentes, como Timely e Sumatra que estendem Java. O Quadro 7.6 mostra uma comparação entre essas linguagens e ISAMadapt.

Aspecto	Klaim	Sumatra	Ponder	ISAMadapt
Suporte à mobilidade	Mobilidade física	Mobilidade de código	Mobilidade de código	Mobilidade lógica e física; Mobilidade do usuário.
Suporte à adaptação	Conectividade	Migração baseada no estado da rede	Baseado em políticas (quem, quando, para onde migrar)	Adaptação de propósito-geral
Expressão da adaptação	Tuplas com localidade	Comando move (migração de threads)	Evento – ação	Comandos, entes e métodos adaptativos, adaptadores
Suporte em execução à adaptação	Runtime da linguagem	Monitoramento da rede	Controle e execução da migração com base na política	ISAMadaptEngine, ISAMcontextService
Linguagem-base	Klaim	Java	Agentes móveis SOMA	J2SE e J2ME
Comandos principais	Enter (abrir conexão) Accept (aceitar conexão) Exit (remover conexão)	move	On ctxElement Subject who Target agent Do action When event	move, clone, reschedule, install, push, discvovery, prefetch, onContext
Modelo de aplicação	Baseado em uma net com node que contém Espaço de Tuplas e processos em execução	Objetos relocáveis	Agentes móveis	Entes com comportamento adaptativo ao contexto
Modelo de interação	Múltiplos Espaço de Tuplas Distribuído	mensagens	mensagens	Espaço de Tuplas Pervasivo, de natureza reativa

Quadro 7.6: Análise Comparativa ISAMadapt e Linguagens para Adaptação

ISAMadapt segue a linha de Klaim, que adicionou características de expressão de conectividade à linguagem Klaim existente. ISAMadapt adiciona expressão de adaptação e consciência de contexto à linguagem Holo, e modifica a semântica de grande parte dos seus comandos. ISAMadapt também possui um alto acoplamento com o ambiente de execução EXEHDA, simulando a relação linguagem-sistema operacional popularizada por C-Unix. Em termos de expressividade e abstrações, pode-se dizer que o ambiente de desenvolvimento ISAMadapt é mais completo, e permite expressar o comportamento das aplicações propostas nestas outras linguagens.

7.7 Resumo: ISAM/ISAMadapt e os Requisitos das Aplicações *Pervasivas*

O Quadro 7.7 resume os requisitos das aplicações *pervasivas* e o modelo de solução adotado na arquitetura ISAM/ISAMadapt.

a aplicação *pervasiva* requer ...	ISAM oferece ...
mobilidade lógica	Entes
mobilidade física (dispositivo, usuário)	Gerenciamento da Mobilidade (EXEHDA)
adaptação na carga do código	Entes adaptativos
adaptação dinâmica	Entes e métodos adaptativos gerenciados pela ISAMadaptEngine
interação assíncrona e anônima	História (espaço de objetos com natureza reativa)
tratamento da desconexão	Desconexão planejada Desconexão em dois níveis: lógica e física
controle da execução	Orientado por políticas de adaptação
conhecimento do ambiente/contexto	Serviços de reconhecimento do contexto - ISAMcontextService
personalização/individualização	Parametrização do contexto
armazenamento *pervasivo* de dados e código	ISAMbda, Ambiente Virtual do Usuário
código portável	Java como linguagem-base
código carregado sob demanda	Mecanismo de class loader de Java, Estrutura minimalista do sistema e Crescimento desta sob-demanda
descoberta de recursos e serviços	Componente `discoverer` do Serviço de Reconhecimento de Contexto
estratégias de adaptação	Entidades de adaptação, Políticas de adaptação e Comandos de adaptação ao contexto
extensibilidade/evolução das estratégias de adaptação	Associação dinâmica de entes aos adaptadores

Quadro 7.7: Requisitos das Aplicações / Oferta da Arquitetura ISAM/ISAMadapt

8 CONCLUSÕES E CONTRIBUIÇÕES

Como os computadores tornam-se mais portáteis, as pessoas desejam acessar informações ou executar tarefas em qualquer lugar, a qualquer hora, com os dispositivos pessoais que carregam junto a si, ou que tenham à disposição no local em que se encontram. Este ambiente delineia o cenário da Computação *Pervasiva*. Tradicionais sistemas distribuídos que assumem um ambiente de execução estacionário não são mais apropriados para cenários extremamente móveis. Adaptar soluções distribuídas construídas com premissas de ambiente estacionário, com conexão permanente e ambiente de execução estável e conhecido, não é indicado.

A mobilidade física impõe restrições inerentes ao ambiente, e estas não podem ser tratadas como exceções, como ocorre nos sistemas distribuídos. O suporte à mobilidade traz um conjunto de novos requisitos e desafios para as aplicações existentes e para as novas aplicações. O núcleo deste desafio está no dinamismo do ambiente conforme o usuário se move para diferentes localizações, usando diferentes terminais, e estando em diferentes contextos e situações.

A pesquisa em ambientes para Computação *Pervasiva* é muito recente, intensificou-se a partir de 2002, e não existe nenhum resultado estabelecido. O tema mobilidade atrai o interesse de pesquisadores, como mostra o crescente número de projetos e a diversidade das propostas de soluções neste campo. Entretanto, observa-se que somente soluções parciais têm sido desenvolvidas, abordando algum aspecto da mobilidade, modificando sistemas existentes ou construindo aplicações *ad-hoc*, a fim de se obter experiência de como e quanto a mobilidade interfere no ambiente computacional distribuído. O projeto ISAM contribui no sentido de fornecer uma visão diferente, a qual integra linguagem e *middleware* de execução para criar um ambiente de Computação *Pervasiva*, como destacado a seguir.

8.1 Conclusões

A arquitetura ISAM foi projetada para atender os requisitos do ambiente de mobilidade lógica e física em âmbito global, e construída com a suposição da existência de uma rede *pervasiva* com suporte à heterogeneidade, conectividade, comunicação e segurança. O cenário da Computação *Pervasiva* exige um sistema de execução complexo, pois deve se ajustar a uma alta dinamicidade de componentes, oriunda de suas propriedades: mobilidade, portabilidade e conectividade.

Sistemas para o desenvolvimento de aplicações para o ambiente *pervasivo* são praticamente inexistentes. Pesquisas muito recentes têm focalizado em requisitos específicos necessários a este ambiente, notadamente o monitoramento de contextos locais. Tais projetos não objetivam a construção de aplicações, mas o gerenciamento do ambiente em si. Um ambiente de suporte à programação de aplicações *pervasivas* hoje

requer, no mínimo, a extensão das linguagens correntes para introduzir noções de contexto e mobilidade.

ISAMadapt é um ambiente de desenvolvimento que aborda de forma integrada os principais aspectos relativos à *pervasividade*. Esta é uma proposta inovadora, uma vez que os demais trabalhos abordam somente algum aspecto da mobilidade e/ou *pervasividade*. A abstração-base da linguagem é o conceito de consciência do contexto. A crença na base da proposta ISAMadapt é que o uso de uma interface uniforme para a adaptação contribui para simplificar o desenvolvimento de aplicações móveis, dinâmicas e sensíveis ao contexto. A linguagem oferece construções que permitem expressar o comportamento adaptativo da aplicação móvel, bem como um ambiente de execução que gerencia o contexto e executa a adaptação, conforme definida e/ou codificada pelo programador.

O objetivo é reduzir a complexidade da produção de aplicações móveis conscientes do contexto, através de abstrações mínimas e construções simples. As abstrações desempenham também um papel de *framework* para o desenvolvimento das aplicações que poderá evoluir para uma metodologia de projeto de aplicações *pervasivas*.

A arquitetura do projeto ISAM permite simular um ambiente *pervasivo*, que ainda não está disponível. Esta simulação permite pensar sobre os requisitos do ambiente, e experimentar com aplicações e com a modelagem de sistemas. A análise realizada para a modelagem da arquitetura ISAM permite concluir que as tecnologias de software disponíveis não são totalmente adequadas para a mobilidade *pervasiva*, pois foram projetadas com premissas de disponibilidade e acesso a recursos que são fixos e conhecidos a priori. A atenção do projetista está na funcionalidade da aplicação, desconsiderando o contexto em que esta executa. Na *pervasividade* o contexto é um elemento de primeira ordem, e deve estar presente na modelagem e ser foco da atenção do projetista.

Acredita-se que as abstrações identificadas e modeladas no ISAMadapt - contexto, adaptadores, comandos e políticas de adaptação - são suficientes para o projeto de aplicações *pervasivas* considerando a tecnologia disponível.

Porém, ao projetar tais abstrações e iniciar o desenvolvimento de aplicações, percebe-se a necessidade do desenvolvimento de um novo paradigma de programação para se atingir a Computação Ubíqua (etapa posterior à Computação *Pervasiva*). Um *insight* conduz a uma visão de programa que não considera uma organização monolítica, com início e fim, mas um programa de longa duração, sendo composto dinamicamente pelas tarefas e necessidades do(s) usuário(s) e adaptando-se ao contexto corrente em que o(s) usuário(s) se encontra(m). Desta forma, programar significaria criar novas tarefas que seriam armazenadas numa base de dados *pervasiva*, e conectadas ao programa, sob demanda, em tempo de execução. O programa seria modelado como um grafo dinâmico de tarefas conscientes do contexto. A linguagem de programação deveria oferecer abstrações para codificar as tarefas e a associação com um contexto. O usuário informaria declarativamente a sua necessidade, e o ambiente de execução comporia o programa (próximas execuções) conforme declarações do usuário, contexto e as tarefas programadas. O contexto deveria ser estendido para situações, as quais consideram grupos de usuários e suas aplicações/tarefas, e não isoladamente como é hoje. Profundas alterações nos sistemas computacionais, especialmente sistemas operacionais e linguagens, seriam necessária para trocar a abordagem baseada em processos para a baseada em tarefas compostas pelo usuário.

Conclui-se, então, que a Computação *Pervasiva* está em sua infância, porém, já está influenciando os sistemas computacionais correntes.

8.2 Contribuições da Pesquisa

A pesquisa no projeto ISAM visa explorar o espaço de projeto em direção à Computação *Pervasiva*. Esta visão inovadora traz desafios e deixa muitos pontos em aberto. O modelo da arquitetura ISAM é intencionalmente amplo, com o objetivo de servir de base a uma larga faixa de futuras pesquisas, permitindo o fortalecimento do grupo e a aglutinação de esforços em torno de um objetivo comum.

Portanto, a primeira contribuição do projeto ISAM não é mensurável e relaciona-se com a inserção da pesquisa em Computação Móvel nas instituições participantes: UFRGS, UFSM, UCPel e UNISINOS, no ano 2000. A disponibilização de uma rede sem fio nessas instituições é um incentivo ao aprofundamento no estudo dos aspectos relativos à computação com mobilidade física, inexistente nessas instituições.

A contribuição científica mais destacada do projeto ISAM é a de fornecer uma arquitetura de software, até então inexistente, para o projeto e suporte à execução de aplicações móveis conscientes do contexto num ambiente *pervasivo*. Esta infra-estrutura simplifica o processo de implementação de aplicações móveis e permitirá a programação de aplicações reais. Estas aplicações, por sua vez, poderão contribuir para a análise da real consequência da mobilidade física nos sistemas de computação, e da efetiva verificação se é possível construir sistemas conscientes de contexto com um comportamento adequado e integrado à vida do usuário móvel.

A pesquisa nos projetos conhecidos para este novo ambiente da computação, como o projeto Aura (www.cs.cmu.edu/~aura), estão em seus primeiros estágios, como ocorre com o projeto ISAM. Alguns elementos para a infra-estrutura *pervasiva* estão presentes nessas pesquisas, mas o desafio de produzir soluções reais, integradas com estes elementos permanece. ISAM é uma contribuição neste sentido.

Criar aplicações móveis conscientes do contexto correntemente é complexo e laborioso. Espera-se que esta situação seja remediada com a criação de uma infra-estrutura que assiste uma variedade de tarefas comuns relativas à consciência do contexto e ao processo de adaptação. A pesquisa em **ISAMadapt** contribui com a simplificação do processo de projeto e programação de aplicações móveis adaptativas através da inserção de abstrações para expressar a consciência do contexto em uma linguagem de programação de propósito-geral. As abstrações identificadas e implementadas no ISAMadapt permitem ao projetista pensar na aplicação em mais alto nível, comparativamente às soluções propostas na literatura. Essas abstrações incluem métricas usadas para guiar a adaptação (estado do elemento de contexto), mecanismos de adaptação com comportamento adaptativo, e algoritmo de decisão automática da adaptação. Desta forma, programas móveis adaptativos ficam mais bem estruturados.

A estratégia de ISAMadapt foi adicionar poucas construções à linguagem-base, e usar uma tradução fonte-a-fonte para o processo de compilação, que une o código a um unificado sistema de execução. Os principais benefícios desta abordagem são: (i) com uma pequena quantidade de suporte da linguagem, a especificação do comportamento adaptativo pode ser separada da especificação da aplicação-base, tornando mais simples de desenvolver, entender e modificar o código adaptativo; (ii) as abstrações podem ser usadas para fornecer um *framework* para adaptação ao contexto, que permite raciocinar sobre adaptação, e escrever código adaptativo, e evoluir para uma metodologia de programação.

Salienta-se que as demais pesquisas abordam o problema de programação através de *frameworks* ou APIs de programação que tratam somente de aspectos específicos como consciência da localização, ou de mecanismos de comunicação. Linguagens de

programação para suportar a descrição e gerenciamento da adaptação em aplicações móveis não são conhecidas.

Especificamente, esta tese contribui para a Ciência da Computação com:

a) uma taxonomia das aplicações móveis adaptativas, identificando suas características e requisitos;
b) a identificação das abstrações e respectiva semântica para expressar um comportamento consciente do contexto no nível de linguagem de programação;
c) um melhor entendimento do contexto das aplicações *pervasivas* através da generalização do processo de reconhecimento e uso do contexto;
d) a identificação de um processo de projeto para construir aplicações móveis conscientes do contexto para o ambiente *pervasivo*, através do *guideline* fornecido pela implementação das abstrações do ISAMadapt.

Neste momento, não se pode avaliar a contribuição em termos de aplicações reais. Esforços estão sendo empreendidos para a implementação de aplicações de utilidade para um usuário móvel. A dificuldade em projetar aplicações reais está no fato de que o ambiente *pervasivo* não está disponível. A concentração dos esforços de pesquisa é na disponibilização de arquiteturas que permitam experimentar com aplicações, como o projeto Aura, Gaia e o ISAM. À medida que novas experiências forem realizadas, poder-se-á identificar as reais necessidades das aplicações para usuários móveis, e detectar se as abstrações propostas aqui são suficientes para implementar a funcionalidade requerida de forma satisfatória.

8.3 Temas em Aberto no ISAMadapt

A arquitetura proposta, ISAM, é abrangente. Este trabalho, ISAMadapt, concentrou-se na questão da expressão da sensibilidade ao contexto no nível de linguagem e ambiente de desenvolvimento. Porém, devido a sua abrangência, a arquitetura ISAM/ISAMadapt apresenta muitos aspectos para serem explorados em trabalhos futuros. O objetivo ao propor esta arquitetura foi justamente o de criar um projeto que oferecesse amplas oportunidades de pesquisas ao longo de um tempo, proporcionando a integração de grupos de várias áreas do conhecimento e fortalecendo a base de conhecimento adquirida. Assim, pode-se identificar alguns temas que merecem atenção na continuidade, destacados a seguir.

8.3.1 Aplicações Reais

Com a ferramenta para programação e o ambiente de execução em operação, pode-se desenvolver as aplicações identificadas e relacionadas no Capítulo 6, de forma a verificar se as abstrações ISAMadapt são suficientemente gerais para as necessidades de formular/identificar/definir/modelar a adaptação ao contexto. Também, com base nestas aplicações pode-se gerar *templates* que simplifiquem a tarefa de projetar/criar novas aplicações.

8.3.2 Serviço de Reconhecimento de Contexto

No momento, o *middleware* `ISAMcontextService` está implementado em um protótipo simples, já que não era o foco principal desta tese, mas um subsídio para execução da adaptação. A questão da distribuição dos componentes, o processo de agregação, interpretação e de predição das informações, e a descoberta de recursos e serviços merecem maior atenção. Formalizar o modelo de contexto utilizado é outra questão em aberto. Estes aspectos são abordados no projeto contextS – um middleware

para desenvolvimento de aplicações conscientes do contexto, aprovado no edital 01/2002 do SEPIn-CNPq-FINEP, a ser desenvolvido no biênio 2003-2004.

8.3.3 Metodologia de Projeto de Aplicações

A modelagem de sistemas *pervasivos*, sobre a qual foi somente apresentado um ponto inicial de discussão, deve ser refinada (REIS et al, 2002). Este foi um trabalho em conjunto com dois doutorandos de Engenharia de Software, orientados pelo Prof. Daltro Nunes, que não teve continuidade devido a questões de escopo das teses dos participantes, e do tempo necessário para concluí-las.

8.3.4 Formalismo das Abstrações

Por questões de tempo e escopo da tese, este aspecto foi omitido. No entanto, foram selecionadas algumas propostas para avaliação futura: MobiS (MASCOLO, 1999) e Ambients (CARDELLI, 2000). A dificuldade encontrada é que essas referências não abordam a questão da mobilidade física nem a questão de contexto – característica natural da Computação *Pervasiva*. Vê-se que o foco atual dos modelos formais recai sobre os aspectos envolvendo a mobilidade lógica – código do programa. Uma exceção é Mobile-Unity (PICCO; ROMAN; McCANN, 2001), equipada com uma variável de localização que pode modelar a migração de um componente ou o movimento de um dispositivo. Seu modelo de comunicação é baseado em memória transientemente compartilhada, adequada ao nosso modelo. Porém, a questão do contexto não é tratada.

8.3.5 Redes Ad-hoc e Computação *Peer-to-Peer*

A arquitetura ISAM foi projetada para redes infra-estruturadas, onde os nodos móveis são gerenciados por componentes da arquitetura que executam na rede fixa. Nos últimos dois anos, as redes *ad-hoc* começaram a ser foco de algumas pesquisas sobre suporte para as aplicações (CAPRA; MASCOLO; EMMERICH, 2002; CUGOLA; PICCO, 2001), ultrapassando o escopo de pesquisa, existente até então, onde predominavam as questões de localização, roteamento e comunicação. Assim, outro ponto passível de exploração é como generalizar as abstrações ISAMadapt para suportar aplicações de redes *ad-hoc*, onde não há gerenciamento centralizado em algum componente da arquitetura, mas este é distribuído entre todos os nodos móveis e/ou fixos.

8.4 Restrições Atuais do ISAMadapt

Além desses temas para futuras pesquisas, existem aspectos relativos ao refinamento de diversas questões tratadas nesta tese com uma abordagem simplificada. As questões mais importantes são destacadas a seguir.

- Desconexão é voluntária. O problema de desconexão involuntária/imprevisível não foi considerado. Este terá de ser tratado com técnicas de tolerância a falhas, avaliadas e adaptadas ao ambiente *pervasivo*, onde recursos são restritos e altamente dinâmicos em disponibilidade e acessibilidade;
- A migração é fraca, implementada através de instanciação remota. A implementação da migração de todo o estado de execução (migração forte) é uma questão a ser explorada, já prevista na implementação da interface `Migrable` do PRIMOS (SILVA, 2003). Atuais implementações da JVM para diversos dispositivos móveis (CLDC) não tem suporte à serialização de objetos,

nem permite criação de *Class Loaders* definidos pela aplicação. Isto dificulta a implementação da migração forte. Outro aspecto a considerar é a eficácia da migração forte, devido ao custo inerente ao processo de salvamento/restauração do estado da execução e os escassos recursos dos dispositivos móveis e da variação no estado da rede sem fio;

- Aplicações são monousuários. Aplicações multiusuários e suas implicações de gerenciamento da adaptação não foram consideradas;
- Diferentes elementos de contexto, e respectiva adaptação, são considerados separadamente. A combinação do estado de diferentes elementos de contexto não foi considerada. Por exemplo, uma aplicação poderia requerer adaptar-se à variação simultânea do estado da rede e do tipo de dispositivo, implicando em 9 combinações de possibilidades. A versão atual do ISAMadapt exige a definição de prioridade para resolver casos de conflito, em que duas ou mais alterações em elementos de contexto ocorram ao mesmo tempo. A combinação de elementos de contexto poderá ser explorada com o serviço de agregação do `ISAMcontextService`;
- Elemento de contexto, e estados correspondentes, tem definição única para a aplicação. ISAMadapt não considera a definição de contexto por ente da aplicação, mas todos os entes que são sensíveis ao mesmo elemento de contexto usarão a mesma definição do elemento;
- Tratamento de erros de execução. Um erro pode ser causado pela indisponibilidade de um recurso/serviço, falha na comunicação, problemas de sincronização, etc... O gerenciamento de erros na execução, para reparos e recuperação deve ser pró-ativo, tratadores executados pelo *middleware*, sob orientação de políticas definidas pela aplicação. Somente uma solução muito preliminar foi definida no ISAMadapt (comando `onError`);
- Alocação de recursos. A solução de adaptação do ISAMadapt considera que existam os recursos mínimos para a execução dos entes, e se adapta a variações no estado dos elementos aos quais a aplicação é sensível. Não foi considerado o caso de não haver esses recursos mínimos, o que exige um sistema de gerenciamento de recursos com alocação negociada e adaptativa – a aplicação informa os níveis de recursos que precisa, e o sistema tenta satisfazê-la;
- Descoberta de recursos e serviços. Este é um aspecto importante para o ISAMadapt, embutido no comando `discovery`, e em outros comandos que necessitam de recursos (lógicos ou físicos), como o ente de sistema `file`. A definição do recurso/serviço em XML, e a resolução (encontrar o recurso que corresponde à descrição definida pela aplicação), a ser armazenada no AVA, devem ser projetadas e implementadas. Um estudo preliminar foi realizado como tema de monografia por Fontoura (2003);
- Segurança – comandos como `install` e `push` exigem critérios de segurança, uma vez que o ambiente computacional do usuário será alterado de forma pró-ativa. Este aspecto foi abordado de forma simplificada. A solução prevista aborda o problema via o conceito de "gerência da reputação", ou seja, na primeira vez que o usuário recebe um comando `install/push` de um nodo, o usuário é questionado se permite a instalação, subseqüentes instalações vindas deste nodo serão consideradas confiáveis pelo usuário.

Pode-se, ao final, concluir que os aspectos relativos à mobilidade estão ainda em processo de desenvolvimento, articulação e entendimento. Criar um ambiente para futuras pesquisas era um dos objetivos quando da criação do projeto ISAM, envolvendo

quatro instituições do RS com potencial de pesquisa em Ciência da Computação, e este foi atingido.

8.5 Publicações do Projeto ISAM/ISAMadapt

8.5.1 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*, Pirenópolis, Goiás, Brasil, set., 2001.

CACIC 2001. Luciano 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*, Santa Cruz, Argentina, oct, 2001.

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

JORNADAS CHILENAS 2001. Edvar Araújo, Iara Augustin, Adenauer Yamin, 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*, Chile. 5-9 Nov. , 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.

8.5.2 Publicações 2002

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). Innsbruck, Austria. 18-21/feb, 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). San Francisco, CA. 4-6/april, 2002, R. Gantenbein and S. Shin Editors, ISCA Publishing, ISBN 1-880843-42-0, p.372-375.

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, Italy, 1-4/july, 2002.

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*, Pasadena California, USA, June.

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). Paris, France, 21-25 oct. 2002.
SBAC/PAD 2002. Adenauer Yamin, Iara Augustin, Jorge Barbosa, Luciano C. da Silva, Rodrigo A. Real, Gerson Cavalheiro, Cláudio F.R. Geyer. Multilevel Colaborative Adaptation in High Heterogeneous Distributed Processing. 14th *SYMPOSIUM ON COMPUTER ARCHITECTURE AND HIGH PERFORMANCE COMPUTING*. Vitória - Brazil, October 28-30.
SCCC 02. AdenauerYamin, Iara Augustin, Jorge Barbosa, Luciano Cavalheiro da Silva, Gerson H. Cavalheiro, Cláudio F.R. Geyer. Collaborative Multilevel Adaptation in Distributed Mobile Applications. XXII *INTERNATIONAL CONFERENCE OF THE CHILEAN COMPUTER SCIENCE SOCIETY*, Atacama, CHILE, 6-8 novembro, 2002.

8.5.3 Publicações 2003

jHPCA 2003. Adenauer Yamin; Iara Augustin; Jorge Barbosa; Luciano C. da Silva; Rodrigo Real; Gerson H. Cavalheiro; Cláudio R. Geyer. Towards Merging Context-aware, Mobile and Grid Computing. *JOURNAL OF HIGH PERFORMANCE COMPUTING APPLICATIONS*. London: Sage Publications. 30p, 2003.
Mobile Computing Handbook. ISAM, joing context-awareness and mobility to building pervasive applications. Iara Augustin, Adenauer Yamin, Luciano C. Silva, Rodrigo Real, Gustavo Frainer, Gerson Cavalheiro, Claudio Geyer. I. Mahgoub and M. Ilyas Ed. Florida. CRC Press. (to be published at December, 2003).
IADIS 2003. Rodrigo Real; Adenauer Yamin; Iara Augustin; Luciano C. da Silva; Cláudio Geyer. Tratamento da incerteza no escalonamento de recursos em *Pervasive Computing*. *IADIS INTERNATIONAL CONFERENCE WWW/INTERNET*, Algarve, Portugal, nov. 2003.

REFERÊNCIAS

ABOWD, G. et al. Cyberguide: a Mobile Context-aware Tour Guide. **Wireless Networks**, New York, v.3, n.5, Oct. 1997.

ACHARYA, S. **Broadcast Disks**: Dissemination-based Data Management for Asymmetric Communication Environment. 1998. Ph.D. Thesis - Brown University, USA. Disponível em: < http://www.cs.umd.edu/projects/bdisk>. Acesso em: dez. 2000.

AHMAD, T. et al. The DIANA Approach to Mobile Computing. In: **Mobile Computing**. New York: Kluwer Academic Press, 1995. Disponível em: <citeseer.nj.nec.com/137108.html>.

ANDRÉ, F.; LE MOÜEL, F. Distribution over Mobile Environment. In: ACM SYMPOSIUM ON APPLIED COMPUTING, SAC, 2000, Como, Italy. **Proceedings…** New York: ACM Press, 2000.

ANDRÈ, F.; SEGARRA, M.-T. A Generic Approach to Satisfy Adaptability Needs in Mobile Environments. In: ANNUAL HAWAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCE, 33., 2000, Maui, Hawaii. **Proceedings…** USA: [S.l.:s.n.], 2000.

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.

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.

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; WORKSHOP EN SISTEMAS DISTRIBUÍDOS Y PARALELISMO, 5., 2001, Puenta Arenas, Chile. **[Anales]**. Puenta Arenas: Universidad de Magallanes, 2001. 1 CD-ROM.

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, 2004. Ainda não publicado.

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 Computer Society, 2002.

AUGUSTIN, I.; YAMIN, A.; GEYER, C. Distributed Mobile Applications with Dynamic Adaptive Behavior. In: INTERNATIONAL CONFERENCE ON COMPUTER AND THEIR APPLICATIONS, CATA, 17., 2002. San Francisco, CA, USA. **Proceedings…**Cary, North Carolina: ISCA Publishing, 2002a.

AUGUSTIN, I.; YAMIN, A.; BARBOSA, J.; SILVA, L.; GEYER, C. Towards Taxonomy for Mobile Applications with Adaptive Behavior. In: INTERNATIONAL SYMPOSIUM ON PARALLEL AND DISTRIBUTED COMPUTING AND NETWORKS, PDCN, 2002, Innsbruck, Austria. **Proceedings...** Calgary, Canada: Acta Press, 2002b.

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, Santa Cruz, Argentina. **[Anales]**. [S.l.:s.n.], 2001. 1 CD-ROM.

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, fev. 2001a.

AUGUSTIN, I. **Acesso aos Dados no Contexto da Computação Móvel**. 2000. 149f. Exame de Qualificação (Doutorado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

AUGUSTIN, I. **pEiffel, uma biblioteca de classes para paralelismo em Eiffel**. 1994. 202 f. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

BAGGIO, A. **Adaptable and Mobile-aware Distributed Objects**. 1999. PhD Thesis. Université Pierre & Marie Curie, Paris. Disponível em: <http://www.sor-inria.fr/~aline>. Acesso em: nov. 2000.

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, 1998, Atlanta, USA. **Proceedings…** New York: ACM Press, 1998.

BANAVAR, G.; BERNSTEIN, A. Software Infrastructure and Design Challenges for Ubiquitous Computing. **Communications of the ACM**, New York, v.45, n.12, Dec. 2002.

BANAVAR, G. et al. Challenges: an Application Model for Pervasive Computing. In: ACM/IEEE INTERNATIONAL CONFERENCE ON MOBILE COMPUTING AND NETWORKING, Mobicom, 6., 2000, Boston, USA. **Proceedings...** New York: ACM Press, 2000.

BARBARÁ, D. Mobile Computing and Databases – a Survey. **IEEE Transactions on Knowledge and Data Engineering**, New York, v.11, n.1, Jan./Feb. 1999.

BARBOSA, J. L. V. **Holoparadigma**: um Modelo Multiparadigma Orientado ao Desenvolvimento de Software Distribuído. 2002. 213f. Tese (Doutorado em Ciência da Computação) – Instituto de Informática,UFRGS, Porto Alegre.

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.

BEADLE, P.; HARPER. B.; MAGUIRE, G.; JUDGE, J. Location Aware Mobile Computing. In: IEEE INTERNATIONAL CONFERENCE ON TELECOMMUNICATIONS, 1997, Melbourne, Australia. **Proceedings…** New York: IEEE Computer Society, 1997.

BELLAVISTA, P.; CORRADI, A.; STEFANELLI, C. Mobile Agent Middleware for Mobile Computing. **IEEE Computer**, New York, v.34, n.3, Mar. 2001.

BETTINI, L.; LORETI, M.; PUGLIESE, R. Modelling Node Connectivity in Dynamically Evolving Networks. In: INTERNATIONAL WORKSHOP ON CONCURRENCY AND COORDINATION, CONCOORD, 2001, Italy. **Proceedings…** Berlin: Springer-Verlag ENTCS, 2001.

BHARGHAVAN, V.; GUPTA, V. A Framework for Application Adaptation in Mobile Computing Environments. In: COMPUTER SOFTWARE AND APPLICATIONS CONFERENCE, 1997, Bethesda, MD. **Proceedings…** [S.l.:s.n.]. Disponível em: < http://timely.crhc.uiuc.edu >. Acesso em: out. 2000.

BHARAT, K. A.; CARDELLI, L. **Migratory Applications**. New York: Digital Equipment Corporation, Systems Research Center, Feb. 1996. (Technical Report, 138).

BLAIR, G. The Role of Open Implementation and Reflection in Supporting Mobile Applications. In: DATABASE AND EXPERT SYSTEMS APPLICATIONS, DEXA, 1998, Vienna, Austria. **Proceedings…** Berlin: Springer-Verlag, 1998. (Lecture Notes in Computer Science, 1460).

BLAIR, G. S. An Architecture for Next Generation Middleware. In: ACM MIDDLEWARE CONFERENCE, MIDDLEWARE, 1998, Lake District, England. **Proceedings...** Berlin: Springer-Verlag, 1998.

BOLLIGER, J.; GROSS, T. A Framework-based Approach to the Development of Network-aware Applications. **IEEE Transactions on Software Engineering**, New York, v.24, n.5, May 1998.

CAPRA, L.; MASCOLO, C. EMMERICH. CARISMA: Context-aware Reflective Middleware System for Mobile Application. **IEEE Transactions on Software Engineering**, New York, v.29, n.3, Mar. 2003.

CAPRA, L.; MASCOLO, C.; EMMERICH, W. XMIDDLE: Information Sharing Middleware for a Mobile Environment. In: ACM INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, ICSE, 2002, Orlando, USA. **Proceedings…** New York: ACM Press, 2002.

CAPRA, L. **Reflective Middleware for Mobile Computing**. Ph.D.Technical Report. July, 2001. Disponível em: <http://www.cs.ucl.ac.uk>. Acesso em: mar. 2003.

CAPRA, L.; MASCOLO, C.; EMMERICH, W. Towards a Mobile Computing Middleware: a Synergy of Reflection and Mobile Code Techniques. In: IEEE WORKSHOP ON FUTURE TRENDS OF DISTRIBUTED COMPUTING SYSTEMS, 8., 2001, Bologna, Italy. **Proceedings…** New York: IEEE Computer Society, 2001a.

CARDELLI, Luca. A Language with Distributed Scope. **Computing Systems**, New York, v. 8, n. 1, Jan. 1995.

CARDELLI, Luca. **Abstractions for Mobile Computing**. New York: Microsoft Research, Aug. 1998. (Techical Report, MSR-TR-98-34). Disponível em: <http://www.citeseer.nj.nec.com/pdf/125645>. Acesso em: abr. 2000.

CARDELLI, Luca; GORDON, A. Mobile Ambients**. Theoretical Computer Science**, New York, v. 240, n. 1, Jan. 2000.

CASTRO, P.; MUNTZ, R. Managing Context for Smart Spaces. **IEEE Personal Communications**, New York, v.7, n.5, May 2000.

CASTILLO, A. et al. Concordia™ as Enabling Technology for Cooperative Information Gathering. In: JAPANESE SOCIETY FOR ARTIFICIAL INTELLIGENCE CONFERENCE, 1998. **Proceedings…** [S.l.:s.n], 1998.

CHALMERS, D.; SLOMAN, M. A Survey of Qualify of Service in Mobile Computing Environments. **IEEE Communications Surveys**, New York, v.2, n.2, Apr. 1999.

CHEN, G. C.; KOTZ, D. **A Survey of Context-aware Mobile Computing Research**. USA: Dartmouth Computer Science, 2000. (Technical Report, TR2000-381).

CHIELE, Rafael. **Aplicações para Dispositivos Móveis com J2ME**: um Estudo de Caso utilizando MIDP/CLDC. 2003. 61 f. Monografia (Especialização em Redes e Sistemas Distribuídos) – Instituto de Informática, UFRGS, Porto Alegre.

CHESS, D.; GROSOF, B.; HARRISON, C.; LEVINE, D.; PARRIS, C. Itinerant Agents for Mobile Computing. **IEEE Personal Communications**, New York, v.2, n.2. Oct. 1995.

CHEVERST, K.; MITCHELL, K.; DAVIES, N. Design of an Object Model for a Context Sensitive Tourist GUIDE. In: INTERACTIVE MOBILE COMPUTING, IMC, 1998, Rostock, Germany. **Proceedings…** [S.l.:s.n.], 1998.

COMP, L.; DOBBING, T. Runtimes Abstractions in the Wireless and Handheld Spaces. **Intel Technology Journal**, [S.l.], v. 7, n.2, Feb. 2003. Disponível em: <http://developer.intel.com/technology/itj/index.html>. Acesso em: ago. 2003.

COMPUTER - Special Issue on Location-aware Computing, New York: IEEE, v.34, n.8, Aug. 2001.

CORRADI, A. et al. Policy Controlled Mobility. In: WORKSHOP ON POLICIES FOR DISTRIBUTED SYSTEMS AND NETWORKS, Policy, 2001, Bristol, UK **Proceedings…** Berlin: Springer-Verlag, 2001. (Lecture Notes in Computer Science, 1995).

CORRADI, A.; LEONARDI, L.; ZAMBONELLI, F. High-level Directives to Drive the Allocation of Parallel Object-Oriented Applications. In: WORKSHOP ON HIGH-LEVEL PROGRAMMING MODELS AND SUPPORTIVE ENVIRONMENTS, 1997, Geneva, Suíça. **Proceedings…** New York: IEEE Computer Society, 1997.

COUDERC, P.; KERMARREC, A.-M. Improving Level of Service for Mobile Users Using Contex-Awareness. In: SYMPOSIUM ON RELIABLE DISTRIBUTED SYSTEMS, SRDS, 18., 1999, Lausanne, Switzerland. **Proceedings…** [S.l.:s.n.], 1999.

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.

CUGOLA, G.; PICCO, G.P. **Peerware**: Core Middleware Support for Peer-to-Peer and Mobile Systems. Italy: Politecnico di Milano, 2001. Disponível em: <www.elet.polimi.it >. Acesso em: mar. 2001.

CUGOLA, G.; GHEZZI, C.; MONGA, M. Coding Different Design Paradigm for Distributed Applications with Aspect-Oriented Programming. In: WORKSHOP SU SISTEMI DISTRIBUITI: ALGORITMI, ARCHITETTURE E LINGUAGGI, 1999, Italy. **Proceedings...** [S.l.:s.n.], 1999.

CYBENKO, G.; CARIPE, W.; MOIZUMI, K.; GRAY, R. Network Awareness and Mobile Agent Systems. **IEEE Communications Magazine**, New York, v.36, n.7, July 1998.

D´AGENT Project HomePage. Dartmouth University. Disponível em: <http://www.cs.dartmouth.edu/~rgray> . Acesso em: dez. 2000.

DAVIES, N.; FRIDAY, A.; WADE, S.; BLAIR, G. Limbo: a Tuple Space Based Platform for Adaptive Mobile Applications. In: INTERNATIONAL CONFERENCE ON OPEN DISTRIBUTED PROCESSING/DISTRIBUTED PLATFORMS, ICODP/ICDP, 1997, Toronto, Canada. **Proceedings…** [S.l.:s.n.], 1997.

DATTA, A.; VANDERMEER, D. E.; CELIK, A.; KUMAR, V. Broadcast Protocols to Support Efficient Retrieval from Databases by Mobile Users. **ACM Transactions on Database Systems**, New York, v.24, n.1, Mar. 1999.

DEWITT, T. et al. **ReMoS**: A Resource Monitoring System for Network-aware Applications. 1997. Pittsburg, USA:Carnegie Mellon University, 1997. (Technical Report, CMU-CS-97-194). Disponível em: <http://www.cs.cmu.edu >. Acesso em: ago. 2000.

DEY, A. K. Understanding and Using Context. **Personal and Ubiquitous Computing**, New York, v.5, n.1, 2001.

DEY, A. K. **Providing Architectural Support for Building Context-aware Applications**. 2000. Doctoral Thesis - Georgia Institute of Technology, Atlanta, USA.

DEY, A. K.; ABOWD, G. D. Towards a Better Understanding of Context and Context-Awareness. In: CONFERENTE ON HUMAN FACTORS IN COMPUTING SYSTEMS, HCI, 14., 2000, UK. **Proceedings…** New York: ACM Press, 2000a.

DEY, Anind K. et al. The Conference Assistant: Combining Context-awareness with Wearable Computing. In: IEEE INTERNATIONAL SYMPOSIUM ON WEARABLE COMPUTERS, ISWC, 3., 1999, San Francisco, CA, USA. **Proceedings…** New York: IEEE Computer Society, 1999.

DRAGOMIRESCU, R. **Dynamic classloading in the kvm**. Rochester, New York:Information Technology Laboratory, Rochester Institute of Technology, Apr. 2000. Technical Report.

EFSTRATIOU, C. et al. Architectural Requirements for the Effective Support of Adaptive Mobile Applications. In: ACM MIDDLEWARE CONFERENCE, MIDDLEWARE, 2000, New York. **Proceedings...** Berlin: Springer-Verlag, 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.

FONTOURA, E. **Estudo da Adequação dos Protocolos de Descoberta de Recursos ao Serviço de Reconhecimento de Contexto da Arquitetura ISAM**. 2003. 68f. Monografia (Especialização em Redes e Sistemas Distribuídos, Edição 2002) – Instituto de Informática, UFRGS, Porto Alegre.

FOX, A. et al. Adapting to Network and Client Variation Using Infrastructural Proxies: Lessons and Perspectives. **Personal Communications**, New York, v.5, n.4, Aug. 1998.

FRIDAY, A.; DAVIES N.; BLAIR, G.S.; WADE, S. P. **Using Tuple Space for Adaptive Mobile Computing**. 1997. UK: Computer Department, Lancaster University, 1997. (Technical Report, MPG-97-03).

FRIDAY, A. **Infrastructure Support for Adaptive Mobile Applications**. 1996. PhD Thesis - Computing Department, Lancaster University, England.

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.

FUGETTA, A.; PICCO, G.P.; VIGNA, G. Understanding Code Mobility. **IEEE Transactions on Software Engineering**, New York, v.24, n.5, May 1998.

GAMMA, E.; JOHNSON, R.; HELM, R, ULISSIDES, J. **Padrões de Projeto, Soluções Reutilizáveis de Software Orientado a Objetos**. Porto Alegre: Bookman, 2000. 364p.

GARLAN, D.; SCHMERL, B. Component-based Software Engineering in Pervasive Computing Environment. In: WORKSHOP ON COMPONENT-BASED SOFTWARE ENGINEERING, ICSE, 4., 2001, Canada. **Proceedings…** Berlin: Springer-Verlag, 2001.

GARLAN, D.; STEENKISTE, P.; SCHMERL, B. Project Aura: Toward Distraction-free Pervasive Computing. **IEEE Pervasive Computing**, New York, v.1, n.3, Sept. 2002.

GELERNTER, D. Generative Communication in Linda. **ACM Computing Surveys**, New York, v.7, n.1, Jan. 1985.

GOMES, A. S. **Implementação de uma Ferramenta de *Groupware* com Suporte a *Awareness*.** 2001. Trabalho de Graduação (Curso Ciência da Computação) - Centro de Tecnologia, UFSM, Santa Maria.

GRAY, R. Agent Tcl: a Flexible and Secure Mobile Agent Systems. In: ANNUAL TCL/TK WORKSHOP, 4., 1996, Monterey, California. **Proceedings...** [S.l.:s.n.], 1996.

GRAY, R.; KOTZ, D.; NOG, S.; RUS, D.; CYBENKO, G. Mobile Agents for Mobile Computing. In: AIZU INTERNATIONAL SYMPOSIUM ON PARALLEL ALGORITHMS/ARCHITECTURES SYNTHESIS, 2., 1997, Japan. **Proceeding…** [S.l.:s.n.], 1997.

GRIMM, R. et al. Programming for Pervasive Computing Environment. In: ACM SYMPOSIUM ON OPERATING SYSTEMS, 18., 2001, Principles, Canada. **Proceedings…**. Berlin:Springer-Verlag,2001.

GRIMM, R. et al. Systems Directions for Pervasive Computing. In: WORKSHOP ON HOT TOPICS IN OPERATING SYSTEMS, HotOS, 8., 2001, Elman, Germany. **Proceedings…** New York: IEEE Computer Society, 2001.

GROSS, T.; STEENKISTE, P.; SUBHLOK, J. Adaptive Distributed Applications on Heterogeneous Networks. In: HETEROGENEOUS COMPUTING WORKSHOP, HCW, 8., 1999, San Juan, Puerto Rico. **Proceedings...** New York: IEEE Computer Society, 1999.

GUPTA, R.; TALWAR, S.; AGRAWAL, D. Jini Home Networking: a Step toward Pervasive Computing. **IEEE Computer**, New York, v.35, n.8, p.34-40, Aug. 2002.

HALLSTEINSEN, S.; LOKEN, T.; NEPLE, T. Adaptivity Support at the Architectural Level. In: WORKSHOP ON SOFTWARE ENGINEERING FOR WEARABLE AND PERVASIVE COMPUTING, ICSE, 2000, Limerick, Ireland. **Proceedings…** Berlin: Springer-Verlag, 2000.

HENRICKSEN, K.; INDULSKA, J.; RAKOTONIRAINY, A. Infrastructure for Pervasive Computing: Challenges. In: WORKSHOP ON PERVASIVE COMPUTING, INFORMATIK, 2001, Vienna, Austria. **Proceedings…** [S.l.:s.n.], 2001.

HENRICKSEN, K.; INDULSKA, J.; RAKOTONIRAINY, A. Modeling Context Information in Pervasive Computing System. In: CONFERENCE ON PERVASIVE COMPUTING, PERVASIVE, 2002, Berlin, Germany. **Proceedings…** Berlin: Springer-Verlag, 2002. (Lectures Notes in Computer Science, 2414).

HESS, C. K.; ROMAN, M.; CAMPBELL, R. Building Applications for Ubiquitous Computing Environments. In: CONFERENCE ON PERVASIVE COMPUTING, PERVASIVE, 2002, Zurich, Switzerland. **Proceedings…**[S.l.:s.n.], 2002.

HEUER, A.; LUBINSKI, A. Data Reduction – an Adaptation Technique for Mobile Environments. In: INTERNATIONAL WORKSHOP ON INTERATIVE APPLICATIONS OF MOBILE COMPUTING, IMC, 1998, Postock, Germany. **Proceedings…** [S.l.:s.n.]. Disponível em: <http://www.rotstock.igd.fhg.de/~imc98/proceedings>. Acesso em: maio 2000.

HONG, J. **Context Fabric**: Infrastructure Support for Context-aware Systems. 2001. Doctoral Proposal - Berkeley University. Disponível em: <www.cs.berkeley.edu>. Acesso em: jun. 2003.

IMIELINSKI, T.; VISWANATHAN, S.; BADRINATH, B.R. Data on Air: Organization and Access. **IEEE Transactions on Knowledge and Data Engineering**, New York, v. 9, n. 3, May 1997.

INTERACTIONS: Special Issue on Designing the PDA of the Future, New York: ACM Press, v. 9, n.1, Jan. 2002.

INTELLIGENT NETWORKS. Tutorial. Disponível em: <www.iec.org/online/tutorials/in>. Acesso em: maio 2003.

JENSEN, F. **An Introduction to Bayesian Networks**. London: UCL Press, 1996. Disponível em: <www.cic.unb.br/~wagner/RedesBayesianas.html>. Acesso em: maio 2003.

JING, J.; HELAL, A.; ELMAGARMID, A. Client-Server Computing in Mobile Environments. **ACM Computing Surveys**, New York, v. 31, n. 2, June 1999.

JING, J.; HUFF, K. Adaptation for Mobile Workflow Applications. In: WORKSHOP ON MODELING AND SIMULATION IN WIRELESS SYSTEMS, 1998, Montreal, Canada. **Proceedings…** [S.l.:s.n.], 1998.

JOSEPH, A. D.; TAUBER, J. A.; KAASHOEK, M. F. Mobile Computing with the Rover Toolkit. **IEEE Transactions on Computers**. Special Issue on Mobile Computing, New York, v.43, n.3, Mar. 1996.

JULIEN, C.; ROMAN, G-C. Egocentric Context-aware Programming in Ad-hoc Mobile Environments. In: ACM SIGSOFT, 2002, Charleston, South Carolina, USA. **Proceedings…** New York: ACM Press, 2002.

JUL, E. et al. Fine Grained Mobility in the Emerald System. **ACM Transactions on Computer Systems**, New York, v.6, n.1, Feb. 1988.

KAMINSKY, A. **JiniME**: Jini Connection technology for Mobile Devices. 2000. Rochester, New York: Information Technology Laboratory, Rochester Institute of Technology. Technical Report. Disponível em: <http://www.csa.rit.edu/~anhinga/ Whitepapers/ JiniMEWhitepaper>. Acesso em: ago. 2003.

KARUNANIDHI, A.; DOERMANN, D.; PAREK, N.; RAUTIO, V. Video Analysis Applications for Pervasive Environments. In: INTERNATIONAL CONFERENCE ON MOBILE AND UBIQUITOUS MULTIMEDIA, 1., 2002, Finland. **Proceedings…** [S.l.:s.n.], 2002.

KATZ, R.H. Adaptation and Mobility in Wireless Information Systems. **IEEE Personal Communications**, New York, v.1, n.1, p.6-17, Jan. 1994.

KELEHER, P.; HOLLINSWORTH, J.; PERKONIC, D. Exploiting Application Alternatives. In: INTERNATIONAL CONFERENCE ON DISTRIBUTED COMPUTING SYSTEMS, ICDCS, 19., 1999, Austin, Texas, USA. **Proceedings…** New York: IEEE Computer Society, 1999.

KICZALES, G. et al. Aspect-oriented Programming. In: EUROPEAN CONFERENCE ON OBJECT-ORIENTED PROGRAMMING, ECOOP, 1997, Finland. **Proceedings…**Berlin: Springer-Verlag, 1997. (Lecture Notes in Computer Science, 1241).

KICZALES, G. et al. An Overview of AspectJ. In: EUROPEAN CONFERENCE ON OBJECT-ORIENTED PROGRAMMING, ECOOP, 15., 2001, Budapest, Hungary. **Proceedings…** Berlin:Springer-Verlag, 2001. Disponível em: <http://www.cs.ubc.ca/ ~gregor>. Acesso em: out. 2002.

KRAUTER, K.; BUYYA, R.; MAHESWARAN, M. Taxomony and Survey of Grid Resource Management Systems for Distributed Computing. **Software – Practice and Experience**, London, v.32, n.2, p.135-164, Feb. 2002.

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.

LEDOUX, T. et al. A Reflective Infrastructure for Coarse-Grained Strong Mobility and its Tool-Based Implementation. In: INTERNATIONAL WORKSHOP ON EXPERIENCES WITH REFLECTIVE SYSTEMS, Reflection, 2001, Japan. **Proceedings…** Berlin: Springer-Verlag, 2001. (Lecture Notes in Computer Science, 2192).

LEE, David G. A Long-term View of Short-range Wireless. **IEEE Computer,** New York, p.39-44, June 2001.

LEI, Zhijun; GEORGABAS, Nicolas. Media Transcoding for Pervasive Computing. In: INTERNATIONAL MULTIMEDIA CONFERENCE, Doctoral Symposium, 2001, Canada. **Proceedings…** New York: ACM Press, 2001. p.459-460.

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, Middleware, 2000, New York, USA. **Proceedings…** Berlin: Springer-Verlag, 2000.

LO, G.; KUNZ, T. A Case Study of Dynamic Application Partitioning in Mobile Computing – an E-mail Browser. In: WORKSHOP ON OBJECT REPLICATION AND MOBILE COMPUTING, ORMC-OOPSLA, 1996, San Jose, California. **Proceedings…** [S.l.:s.n.], 1996.

LOWEKAMP, B. et al. A Resource Query Interface for Network-aware Applications. IEEE SYMPOSIUM ON HIGH-PERFORMANCE DISTRIBUTED COMPUTING, 7., 1998, Chicago, USA. **Proceedings…** New York: IEEE Computer Society, 1998.

LU, S.; BHARGHAVAN, V.; LEE, K-W. Adaptive Service in Mobile Computing Environments. In: INTERNATIONAL WORKSHOP ON QUALITY OF SERVICE, Service, 1997, New York, NY. **Proceedings…** [S.l.:s.n.], 1997.

MASCOLO, C. MobiS: Specification Language for Mobile Systems. In: INTERNATIONAL CONFERENCE ON COORDINATION LANGUAGE AND MODELS, Coordination, 3., 1999, Amsterdan, Netherland. **Proceedings…** Berlin: Springer-Verlag, 1999. (Lecture Notes in Computer Science, 1594).

McILHAGGA, M.; LIGHT, A.; WAKEMAN, I. Towards a Design Methodology for Adaptive Applications. In: INTERNATIONAL CONFERENCE ON MOBILE COMPUTING AND NETWORKING, MOBICOM, 1998, Dallas, Texas, USA. **Proceedings…** Berlin: Springer-Verlag, 1998.

MIHAILESCU, P.; KENDALL, E. A Mobile Development an Agent Platform for Mobile Device using J2ME. In: EVOLVE CONFERENCE, 2001, Sidney, Australia. **Proceedings…** [S.l:s.n.], 2001.

MILLER N.; STEENKISTE, P. Collecting Network Status Information for Network-aware Applications. In: INFOCOM, 2000, Tel Aviv. **Proceedings…** [S.l.:s.n.], 2000.

MUNIR, S. **Active Networks** – a Survey. 2000. Disponível em: <http://www.cis.ohio-state.edu/~jain/cis788-97/active-nets.html>. Acesso em: ago.2000.

MURPHY, A. L.; PICCO, G. P.; ROMAN, G.-C. LIME: Linda Meets Mobility. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, ICSE, 1999, Los Angeles, USA. **Proceedings…** Berlin: Springer-Verlag, 1999.

NOBLE, B. System Support for Mobile, Adaptive Applications. **IEEE Personal Computing Systems**, New York, v.7, n.1, p. 44-9, Feb. 2000.

NOBLE, B. **Mobile Data Access**. 1998. PhD. Thesis - School of Computing Science, Carnegie Mellon University, Pittsburg, USA. Disponível em: <www.cs.cmu.edu/afs/cs/projects/coda/web/docdir/bnoble-thesis>. Acesso em: jun. 2000.

NORMAN, D. A. **The Invisible Computer**. Cambridge, MA: MIT Press, 1998.

OKAMURA, H.; ISHIKAWA, Y. Object Location Control Using Meta-level Programming. In: EUROPEAN CONFERENCE ON OBJECT-ORIENTED PROGRAMMING, ECCOP, 1994, Bologna, Italy. **Proceedings…** Berlin: Springer-Verlag, 1994. (Lecture Notes in Computer Science, 821).

ORAM, A. **Peer-to-peer**: Harnessing the Power of Disruptive Technologies. New York:O'Reilly & Associates, 2001.

PASCOE, J.; MORSE, D.; RYAN, N. Developing Personal Technology for the Field. **Personal Technologies**, New York, v.2, n.1, Mar. 1998.

PEER-TO-PEER Working Group. 2002. Disponível em: <http://www.peer-to-peerwg.org>. Acesso em: nov. 2001.

PEINE, H.; STOLPMANN, T. The Architecture of the Ara Platform for Mobile Agents. In: INTERNATIONAL WORKSHOP ON MOBILE AGENTS, 1, 1997. Berlin, Germany. **Proceedings…** [S.l.:s.n.], 1997. Disponível em: <http://www.uni-kl.dc/AG-Nehmer/projekte/Ara/index-e-html>. Acesso em: dec.1999.

PHAM, V. A.; KARMOUCH, A. Mobile Software Agents: an Overview. **IEEE Communications Magazine**, New York, v.36, n.7, July 1998.

PICCO, G. P.; BUSCHINI, M. L. Exploiting Transiently Tuple Spaces for Location Transparent Code Mobility. In: INTERNATIONAL CONFERENCE ON COORDINATION MODELS AND LANGUAGES, Coordination, 5., 2002, York, UK. **Proceedings…** Berlin: Springer-Verlag, 2002, p.258-273. (Lecture Notes in Computer Science, 2315).

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.

PICCO, G. P.; ROMAN, G. –C.; McCANN, P. J. Reasoning about Code Mobility in Mobile Unity. **ACM Transactions on Software Engineering and Methodology**, New York, v.10, n.3, July 2001.

PICCO, G. P. mucode: Lightweight and Flexible Code Toolkit. In: INTERNATIONAL WORKSHOP ON MOBILE AGENTS, MA, 2., 1998, Stuttgart, Germany. **Proceedings…** New York: Springer-Verlag, 1998, p.160-171. (Lecture Notes in Computer Science, 1477).

PITOURA, E. **Software Models for Mobile Wireless Computing**. Jyvaskyla, Finland: Summer School, Aug. 1998. Disponível em: <www.cs.uoi.gr/~pitoura>. Acesso em: maio 2001.

RAKOTONIRAINY, A.; GROVES, G. Resource Discovery for Pervasive Environment. In: INTERNATIONAL SYMPOSIUM ON DISTRIBUTED OBJECTS AND APPLICATION, DOA, 4., 2002, USA. **Proceedings…** New York: IEEE Computer Society, 2002.

RANGANATHAN, A.; CAMPBEL, R. H.; MAHAJAN, A. Conchat: a Context-aware Chat Program. **IEEE Pervasive Computing**, New York, n.1, v.3, July-Sept. 2002.

RANGANATHAN, M.; ACHARYA, A.; SALTZ, J. Sumatra: a Language for Resource-aware Mobile Programs. In: INTERNATIONAL WORKSHOP ON MOBILE OBJECT SYSTEMS, MOS, 1996, Linz, Austria. **Mobile Objects Systems**: Towards the Programmable Internet: selected presentations and invited papers. Berlin: Springer-Verlag, 1997. (Lecture Notes in Computer Science, 1222).

RANGANATHAN, M.; ACHARYA, A.; SHARMA, S.; SALTZ, J. Network-aware Programs. In: ANNUAL TECHNICAL CONFERENCE, USENIX, 1997, California, USA. **Proceedings...** [S.l.:s.n.], 1997a.

REAL, R.; YAMIN, A.; AUGUSTIN, I.; SILVA, L. C.; GEYER, C. Tratamento da Incerteza no Escalonamento de Recursos em *Pervasive Computing*. In: IADIS INTERNATIONAL CONFERENCE WWW/INTERNET, 2003, Algarve, Portugal. **Proceedings...** [S.l.:s.n.], 2003.

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.

ROMAN, M. **An Application Framework for Active Space Applications**. 2003. Ph.D. Thesis - University of Illinois at Urbana-Champaign, Illinois, USA. Disponível em: <www.cs.uiuc.edu/gaia>. Acesso em: abr. 2003.

ROMAN, M. et al. Gaia: a Middleware Infrastructure to Enable Active Spaces. **IEEE Pervasive Computing**, New York, v.1, n. 4, Dec. 2002.

ROMAN, G-C.; PICCO, G. P.; MURPHY, A. L. A Software Engineering Perspective on Mobility. In: FINKELSTEIN A.C.W. (Ed.). **Future of Software Engineering**. New York: ACM Press, 2000.

SAHA, D.; MUKHERJEE, A. Pervasive Computing: a Paradigm for the 21st Century. **IEEE Computer**, New York, v.36, n.3, p.25-31, Mar. 2003.

SATYANARAYANAN, M. Pervasive Computing: Vision and Challenges. **IEEE Personal Communications**, New York, v.4, n.8, Aug. 2001.

SATYANARAYANAN, M. Fundamental Challenges in Mobile Computing. In: ACM SYMPOSIUM ON PRINCIPLES OF DISTRIBUTED COMPUTING, 1996, Philadelphia, Pennsylvania, USA. **Proceedings…** Berlin: Springer-Verlag, 1996.

SATYANARAYANAN, M. Mobile Information Access. **IEEE Personal Communications**, New York, v.3, n.2, Feb. 1996a.

SCHAFFER, A. E. **Descoberta de Recursos no ambiente ISAMpe**. 2004. Proposta de Dissertação (Curso de Ciência da Computação) - Instituto de Informática, UFRGS, Porto Alegre. Em preparação.

SCHILIT, B.; NORMAN, A.; WANT, R. Context-aware Computing Applications. In: IEEE WORKSHOP ON MOBILE COMPUTING SYSTEMS AND APPLICATIONS, 1994, Santa Cruz, CA, USA. **Proceedings…** New York: IEEE Computer Society, 1994.

SILVA, L. C. **Primitivas para Suporte à Distribuição de Objetos Direcionados à *Pervasive Computing***. 2003. 120f. Dissertação (Mestrado em Ciência da Computação) - Instituto de Informática, UFRGS, Porto Alegre.

STEENKISTE, P. Adaptation Models of Network-aware Distributed Computations. In: INTERNATIONAL WORKSHOP ON COMMUNICATIONS AND ARCHITECTURAL SUPPORT FOR NETWORK-BASED PARALLEL COMPUTING, CANPC, 3., 1999, Orlando, Florida, USA. **Proceedings…** Berlin: Springer-Verlag, 1999. (Lecture Notes in Computer Science, 1602).

SUN MICROSYSTEMS. **A Survey of J2ME Today**. Nov. 2002. Disponível em: <http://wireless.java.sun.com/getstart/articles/surveys>. Acesso em: dec. 2002.

TENNENHOUSE, D. Proactive Computing. **Communications of the ACM**, New York, v.43, n.5, p.43-50, May 2000.

TERRY, D.; THEIMER, M.; PETERSEN, K.; DEMERS, A.; SPREITZER, M.; HAUSER, C. Managing Update Conflict in Bayou, a Weakly Connected Replicated Storage Systems. In: MILOJICIC, D.; DOUGLIS, F.; WHELER, R. (Ed.). **Mobility, Processes, Computers and Agents**. New York: ACM Press, 1999.

TSENG, Y.; WU, S.; LIAO, W.; CHAO, C. Location Awareness in Ad Hoc Wireless Mobile Networks. **IEEE Computer**, New York, v.34, n.6, June 2001.

THORN, T. Programming Languages for Mobile Code. **ACM Computing Surveys**, New York, v.29, n.3, Sept. 1997.

VIGNA, G. **Mobile Code, Technologies, Paradigms, and Applications**. 1998. Tesi di Dottorato - Politecnico di Milano, Italy.

WAKEMAN, I.; ORMSBY, A.; MCIHAGGA, M. Signaling in a Component Based World. In: OPEN ARCHITECTURES FOR SIGNALLING, 1., 1998, San Francisco, CA, USA. **Proceedings…** New York: IEEE Computer Society, 1998.

WANG, J. **A Proxy Server Infrastructure for Adaptive Mobile Applications**. 1999. Master Thesis - Carleton University, Ottawa, Canada.

WEISER, M. Some Computer Science Problems in Ubiquitous Computing. **Communications of the ACM**, New York, v.36, n.6, July 1993.

WEISER, M. The Computer of the 21st Century. **Scientific American,** New York, v.265, n.9, Sept. 1991.

WELLING, G.; BADRINATH, B. R. An Architecture for Exporting Environment Awareness to Mobile Computing Applications. **IEEE Transactions on Software Engineering**, New York, v.24, n.5, May 1998.

WHITE, J. E. **Telescript Technology: Mobile Agents**. 1996. White paper - General Magic. Disponível em: <www.genmagic.com/Telescript/Whitepapers>. Acesso em: ago. 2000.

WINOGRAD, T. Architectures for Context. In: HUMAN COMPUTER INTERACTION. Special Issue on Context-aware Computing, HCI, 2001, Patras, Greece. **Proceedings…** Berlin: Springer-Verlag, 2001.

YAMIN, A. **Arquitetura para um Ambiente de Execução Direcionado às Aplicações Móveis Conscientes do Contexto em um Ambiente de *Pervasive Computing***. 2004. Proposta de Tese (Doutorado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre. Em preparação.

YAMIN, A.; AUGUSTIN, I.; BARBOSA, J.; SILVA, L.; REAL, R.; CAVALHEIRO, G.; GEYER, C. Towards Merging Context-aware, Mobile and Grid Computing. **Journal of High Performance Computing Applications**, London, v.17, n.2, p.191-203, 2003.

YAMIN, A.; AUGUSTIN, I.; BARBOSA, J.; SILVA, L.; REAL, R.; CAVALHEIRO, G.; GEYER, C. Multilevel Collaborative Adaptation in High Heterogeneous Distributed Processing. In: SYMPOSIUM COMPUTER ARCHITECTURE AND HIGH PERFORMANCE COMPUTING, SBAC, 14., 2002, Vitória, ES, Brazil. **Proceedings…** Vitória: SBC, 2002.

YAMIN, A.; AUGUSTIN, I.; BARBOSA, J.; GEYER, C. ISAM: a Pervasive View in Distributed Mobile Computing. In: NETWORK CONTROL AND ENGINEERING FOR QOS, SECURITY AND MOBILITY, 2002, Paris, France. **Proceedings…** New York: IEEE Computer Society, 2002a.

YAN, H.; SELKER, T. Context-aware Office Assistant. In: INTELLIGENT USER INTERFACES, 2000, New Orleans, LA, USA. **Proceedings…** [S.l.:s.n.], 2000.

YARVIS, M.; REIHER, P.; POPEK, G. J. Conductor: a Framework for Distributed Adaptation. In: WORKSHOP ON HOT TOPICS IN OPERATING SYSTEMS, hotOS, 7., 1999, Rio Rico, Arizona, USA. **Proceedings…** Berlin: Springer-Verlag, 1999.

ZACHARIADIS, S.; CAPRA, L.; MASCOLO, C.; EMMERICH, W. XMIDDLE: Information Sharing Middleware for a Mobile Environment. In: INTERNATIONAL CONFERENCE OF SOFTWARE ENGINEERING, ICSE, 24., 2002, Orlando, Florida, USA. **Proceedings…** Berlin: Springer-Verlag, 2002.

ZAMBONELLI, F.; CABRI, G.; LEONARDI, L. MARS: a Programmable Coordination Architecture for Mobile Agents. **Internet Computing**, New York, v.4, n.4, July-Aug. 2000.

ZENEL, B.; DUCHAMP, D. A General Purpose Proxy Filtering Mechanism Applied to the Mobile Environment. In: ACM/IEEE INTERNATIONAL CONFERENCE ON MOBILE COMPUTING AND NETWORKING, MOBICOM, 3., 1997, Budapest, Hungary. **Proceedings…** Berlin: Springer-Verlag, 1997.

APÊNDICE A ORGANIZANDO CONCEITOS

O conceito de Computação Móvel ainda não é consenso, e tem evoluído no início dessa década. Não existe um *framework* conceitual para a mobilidade hoje. Diferentes aspectos da mobilidade são tratados em vários projetos que são desenvolvidos para diferentes domínios e focalizam diferentes pontos de vista relativos aos sistemas. Como os conceitos e definições estão ainda se delineando (não existe um corpo de conhecimento estável), este Capítulo aborda a visão da autora sobre a área de Computação Móvel, conceitos e definições sob os quais se desenvolve esta tese. Neste texto, faz-se uma revisão informal dos conceitos relativos à mobilidade, apresentam-se os conceitos fundamentais relativos aos sistemas de comunicação e informação, e serviços oferecidos por esses sistemas.

A.1 Cenários da Mobilidade

Vê-se que este termo computação móvel tem sido usado com diferentes significados. O ponto em comum é que envolvem alguma forma de mobilidade (AUGUSTIN, 2000). Neste trabalho, adota-se o conceito "Computação Móvel é a computação distribuída onde os elementos computacionais podem trocar de localização durante o curso da computação". Estes elementos podem ser: o equipamento e a rede (hardware); o código, os dados, o programa em execução (software); e o usuário (cliente móvel). Esta conceituação é ampla, e dependendo da combinação dos elementos com a propriedade de mobilidade, existem vários cenários possíveis de interesse da Computação Móvel atual.

Os principais cenários atuais, que envolvem o conceito de mobilidade dos elementos computacionais, são:

a) Computação sem fio (*wireless computing*) – o usuário com seu dispositivo móvel pode se deslocar dentro da área de acesso da rede sem fio local. A alteração mais relevante se dá na camada de transporte da rede, as aplicações executadas podem ser as mesmas da rede cabeada;
b) Computação nômade (*nomadic computing / Palm computing*) – o usuário com seu computador portátil (PDA) executa suas aplicações de administração pessoal, este ocasionalmente conecta-se com a rede via conectores de rede fixa;
c) Mobilidade de código (*mobile computation / mobile agents*) – o software é móvel. Dependendo do elemento do software (data, código, estado de execução) que pode se mover, este pode ser classificado como mobilidade fraca – somente o código e dados são migrados, ou mobilidade forte – o estado da execução é migrado. Um estudo sobre esta temática foi realizado na tese de doutorado de Vigna (VIGNA, 1998);

d) Computação *pervasiva* (*pervasive computing*) – fornece o cenário mais geral da mobilidade, onde todos os elementos computacionais podem ser móveis. Permite a mobilidade lógica e física enquanto mantém a conectividade global.

Esses cenários não são excludentes; em geral, o ambiente móvel pode ser projetado com uma combinação de cenários. Por exemplo: agentes móveis que executam na rede fixa, ou agentes móveis que executam em clientes móveis (PDA's).

A.1.1 Linha Evolutiva da Mobilidade

Para uma melhor classificação, pode-se enquadrar esses cenários em uma linha evolutiva do conceito de Computação Móvel. A Figura A.1 esboça uma seqüência evolutiva e as principais áreas da computação que contribuem em cada fase dessa evolução. Esta organização baseia-se na proposta de uma taxonomia de Satyanarayanan (2001), que foi modificada para acomodar a visão da autora, principalmente na separação entre Computação *Pervasiva* e Computação Ubíqua (Satyanarayanan considera somente como linha evolutiva 1-2-3, e *Pervasive Computing* como sinônimo de *Ubiquitous Computing*).

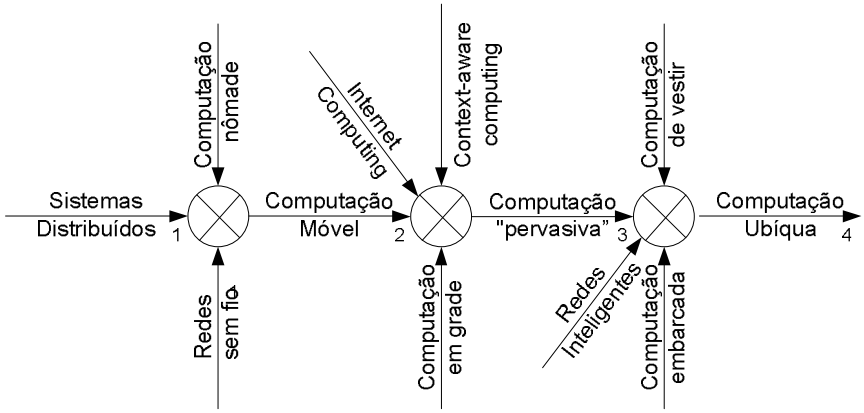


Figura A.1: Evolução do Conceito de Mobilidade e Computação

Considera-se que o corpo de conhecimento para se chegar à computação ubíqua, como proposto originalmente por Mark Weiser (1991), está sendo construído a partir das soluções empregadas em muitas áreas da computação, entre elas Sistemas Distribuídos, Computação Móvel e Computação *Pervasiva*. Pode-se dizer que existe um movimento de convergência das pesquisas e tecnologias para a computação ubíqua. Todos estas áreas são, hoje, campos ativos de pesquisa, em constante evolução, e cujas soluções poderão contribuir para a construção da computação invisível, pró-ativa e integrada na vida do usuário.

Sistemas distribuídos contribuem com fundamentos para comunicação e acesso a informação remota, tolerância a falhas, alta disponibilidade, segurança (SATYANARAYANAN, 2001). Computação Nômade contribui com portabilidade, foco nas atividades do usuário, e aplicações leves, com pouco consumo de recursos. Computação Sem Fio, através das redes móveis, criadas com tecnologia sem fio, permitem o acesso e disponibilidade de conexão em qualquer lugar. Também, o acesso à informação personalizada, através de banco de dados móveis (AUGUSTIN, 2000).

A **Computação Móvel**, nesta linha de evolução, é o **momento tecnológico atual** da computação, a qual une a portabilidade da computação nômade com a acessibilidade e conectividade das redes sem fio. Busca, essencialmente, o acesso a informações em qualquer lugar, usando vários dispositivos e conectividade local. No passado, o

movimento do usuário causava interrupção do serviço da rede, isto agora não é mais suportado. Um dos primeiros exemplos de tecnologia de acesso móvel à informação foi o protocolo WAP (*wireless Application Protocol*) para acesso a páginas Web a partir de dispositivos de baixa capacidade, como celulares (AUGUSTIN, 2000).

Aplicações-protótipo com funcionalidades diversas baseadas em informações de algum aspecto do contexto (*Context-aware Computing*) estão sendo desenvolvidas: considerando a localização do usuário, como Guide (CHEVERST; MITCHELL; DAVIES, 1998); a escalabilidade de usuários, como o *chat* adaptalk (RANGANATHAN; ACHARYA; SHARMA; SALTZ, 1997); e informações contextuais para enriquecer a comunicação eletrônica, como conchat (RANGANATHAN; CAMPBEL; MAHAJAN, 2002). Segundo Chen e Kotz (2000) poucas aplicações usam outras informações, além da localização (*location-aware applications*), para fornecer conteúdo ou comportamento diferenciado. A contribuição desta área é em como obter informações de contexto, estático e dinâmico, e como interpretá-lo. Como a área de sistemas adaptativos (*adaptive systems*) está intrinsicamente relacionada a esta, foi considerada inserida nela. Estratégias de adaptação são a contribuição da área para o comportamento *pervasivo* das aplicações.

A **Computação *Pervasiva* é o futuro próximo**, quando uma rede de conexão móvel global (rede *pervasiva*) estará disponível, permitindo a mobilidade do código e dados (lógica) acompanhando a mobilidade do usuário e *hosts* (física), de forma que a computação possa estar espalhada, em todo o lugar. Henricksen, Indulska e Rakotonirainy (2001) fazem um levantamento dos desafios atuais para construir-se uma infra-estrutura de suporte à Computação *Pervasiva*, e classificou-os em quatro classes relativas a dispositivos, usuários, componentes de software e interfaces de usuário.

Hoje, o exemplo mais claro de *pervasibilidade* é o fornecido pela *Internet Computing*, que contribui com uma plataforma comum de software, agentes móveis/mobilidade de código, e sistemas adaptativos aos usuários. A diferença básica entre elas é no objetivo, a Computação *Pervasiva* visa atender ao usuário e suas atividades, permitindo a este ter em qualquer lugar que esteja o seu ambiente computacional (aplicações, dados, preferências e configurações), enquanto que na WWW há uma impessoalidade, todos (ou grupos) acessam os mesmos dados, visando à disponibilidade de informações. A computação em grade (*Grid Computing*) caminha no sentido de permitir a execução de aplicações de forma espalhada, contribui com a infra-estrutura de gerenciamento da distribuição em larga escala, e do gerenciamento dos recursos. A computação consciente do contexto contribui com soluções para a aquisição de informações de contexto e adaptação à variação deste. Segundo Satyanarayanan (2001), Computação *Pervasiva* será uma fértil fonte de problemas de pesquisa nos sistemas computacionais por muitos anos.

Em termos de redes móveis, além do desenvolvimento das redes sem fio, estão surgindo conceitos que dinamizam a estrutura organizacional das redes atuais. Redes Inteligentes (IN – *Intelligent Network*) é uma nova proposta de redes sem fio que se auto-organizam e gerenciam o tráfego de dados sem a interferência humana (Intelligent Networks, 2003). Outro conceito é o de redes ativas (*Active Networks*) na qual recursos internos das redes, tipicamente roteadores, tornam-se programáveis. Nestas redes, alem do tráfego de dados tradicional, programas podem ser injetados pelo usuário, os quais serão interpretados pelos nós da rede, e executados sob o fluxo de dados que trafega (MUNIR, 2000). Computação embarcada, e computação de vestir (*Wearable Computing*) são tecnologias de automação que contribuem com a inserção de novos dispositivos, com funcionalidades específicas, que auxiliam as pessoas a desenvolver suas atividades diárias.

Acredita-se que a computação ubíqua será uma realidade quando todas essas tecnologias convergirem e estiverem tão presente na vida do usuário que a computação se torna invisível a ele. A pró-atividade do sistema é importante para este ambiente, porém constitui-se num dos seus maiores desafios: saber o que o usuário quer para lhe fornecer. Os sistemas atuais são pobres em descobrir a intenção do usuário (SATYANARAYANAN, 2001).

O Quadro A.1 traça um comparativo das características e requisitos dos sistemas nas diversas fases da Computação Móvel.

Características e requisitos	Computação Nômade	Computação Móvel	Computação *Pervasiva*	Computação Ubíqua
Período preponderante	Década 90	Início 2000	Futuro próximo	Futuro (indeterminado)
Domínio de aplicações	Administração pessoal	Sistemas de informação	Centrado no usuário e suas atividades	Gerenciamento pró-ativo dos usuários
acesso à rede	Discado ou cabeado	Conexão sem fio (Wi-fi, rede celular)	Conexão sem fio (IEEE 802.11, bluetooth)	Novas tecnologias de conexão sem fio
Tipo de conexão	ocasional	Sem fio, local, intermitente	Sem fio, global e permanente	Sem fio, total, permanente
Plataformas	PalmOS, PocketPC, SymbianOS, Linux, ...	Celulares, PalmOS, Linux, PocketPC, SymbianOS,...	independente	independente
dispositivos	Computadores de mão tradicionais	Celulares, computadores de mãos, com acesso sem fio	independente	Integrados e pró-ativos
interface	Menus simples, reduzida ao campo do visor, e entrada textual	Menus simples, Reduzida ao campo do visor, e entrada textual	Adaptativa ao contexto (texto, voz, gráfica,...)	Intuitiva, Pró-ativa
restrições	Consumo de energia, capacidade de processamento e armazenamento	Consumo de energia, capacidade de processamento e armazenamento	Adaptabilidade para vencer restrições dos dispositivos móveis	Uso de vários dispositivos com funcionalidades diferentes
adaptabilidade	Aos recursos físicos	Aos recursos lógicos e físicos	Ao contexto	Ao contexto
gerência de recursos	Alocação	Alocação e negociação QoS	Dinâmica, adaptativa	Dinâmica, adaptativa, autogerência
instalação de softwares	Residente	Residente	Dinâmica, sob demanda e/ou pró-ativa	Dinâmica, sob demanda e/ou pró-ativa
linguagens para programação	Tradicionais, versão reduzida: C, Java.	Tradicionais, versão reduzida: C, Java, Toolkits para acesso sem fio	Novas linguagens, sensíveis ao contexto, programação para troca	Novo paradigma: baseado na combinação de tarefas
plataforma de desenvolvimento	PC. Emuladores/ simuladores, APIs, Desenvolvimento cruzado, sincronização PC-dispositivo	PC. Emuladores/ simuladores, APIs, toolkits, Desenvolvimento cruzado, sincronização PC-dispositivo	PC. Emuladores/ simuladores, APIs, toolkits, Desenvolvimento cruzado, sincronização PC-dispositivo	indefinido

Quadro A.1: Quadro Comparativo das Características dos Sistemas Móveis

A.1.2 Propriedades da Computação Móvel

A principal diferença que caracteriza uma aplicação da Computação Móvel é a existência de três propriedades: portabilidade, mobilidade e conectividade (AUGUSTIN, 2001a). A **portabilidade** é fornecida pelo dispositivo móvel, o usuário portando este equipamento pode deslocar-se para qualquer lugar. A **conectividade** pode ser permanente, usando a rede fixa, ou intermitente, usando a rede sem fio. A **mobilidade** pode ser tanto física (do usuário e equipamentos de hardware) quanto lógica (da computação – dados, código e estado de execução).

Os cenários acima podem ser definidos em termos de **mobilidade física** como ilustrado na Figura A.2.

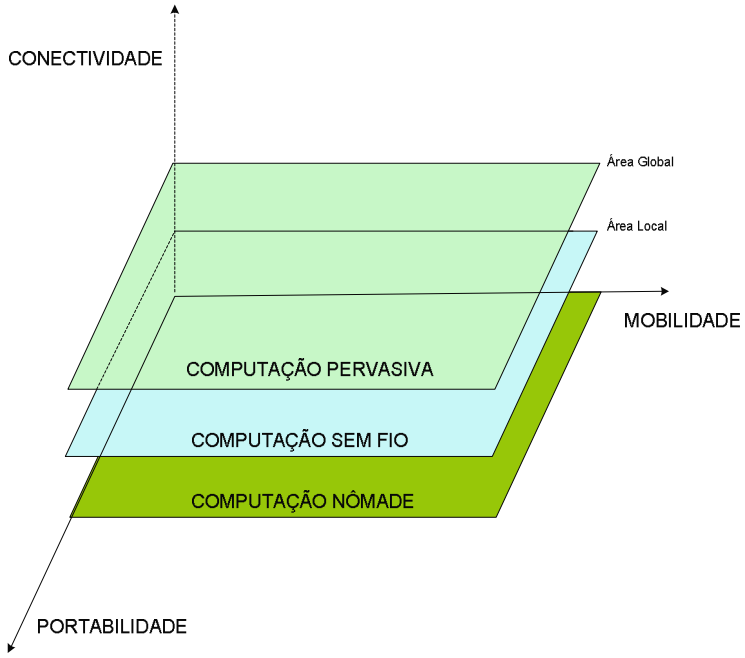


Figura A.2: Cenários e Propriedades da Mobilidade Física

A portabilidade é a propriedade que caracteriza a **Computação Nômade**. Nesta, a mobilidade está associada ao dispositivo, portado pelo usuário, que se desloca (mobilidade do *host*), mas como as aplicações são residentes no dispositivo esta não afeta o sistema. A conectividade, neste caso, é em geral via acesso discado/cabeado, é permanente e cada deslocamento do dispositivo requer uma nova conexão.

Nas aplicações da **Rede sem fio**, a mobilidade do usuário e a conectividade à rede são limitadas à área local de acesso da rede. As aplicações, em geral, são as mesmas da rede fixa, podendo apresentar aplicações com mobilidade lógica (agentes móveis que trafegam também na rede sem fio).

Por definição, nas aplicações *pervasivas* a mobilidade está associada ao usuário (mobilidade física), e à computação (mobilidade lógica) em uma escala global. A conectividade é intermitente, via rede sem fio global (*wide area)*. A mobilidade é explorada em sua forma máxima: o usuário pode carregar seu próprio dispositivo móvel ou usar qualquer um que esteja disponível no lugar em que se encontra. O ambiente computacional do usuário o seguirá e estará disponível em qualquer lugar a qualquer tempo. Este é um cenário de uma nova dimensão para as futuras redes de computação e de comunicação.

Nas aplicações de **mobilidade de código** e agentes móveis (mobilidade lógica), a mobilidade está associada ao elemento de software (dados, código ou estado de

execução). A conectividade é permanente e global, pois foram projetadas originalmente para uso na rede fixa (*Internet Computing*).

Considerando a mobilidade lógica no ambiente da Computação Móvel, o grau de presença da mobilidade da computação (*mobile computation*) pode variar em cada cenário, exceto na computação nômade que não oferece mobilidade de código. Trabalhos recentes, como o do projeto ISAM (www.inf.ufrgs.br/~isam) associam mobilidade lógica com mobilidade física visando o ambiente computacional *pervasivo*.

Inspirados na taxonomia de abordagens para o projeto de aplicações com mobilidade de código apresentada por Vigna (1998), categorizam-se as estratégias de mobilidade de código em:

a) *pull* – a aplicação requisita o código sob demanda;
b) *push* – o sistema envia o código para avaliação remota; e
c) agentes móveis – a aplicação decide migrar sua computação.

O ambiente *pervasivo* requer a disponibilidade destas três estratégias, cada uma delas útil em uma dada situação. A maioria dos sistemas disponíveis somente disponibiliza uma destas formas de mobilidade. Estas são requeridas para atender os requisitos de implementação da semântica siga-me (*follow-me*) das aplicações *pervasivas*.

O Quadro A.2 mostra um quadro comparativo entre cenários da mobilidade e suas propriedades.

Portabilidade, mobilidade e conectividade inserem **restrições** no ambiente móvel. O computador portátil é pequeno, leve e para tal requer recursos de energia de baixo poder. Como conseqüência, computadores portáteis têm restrições no tamanho da memória, na capacidade de armazenamento, no poder computacional e na interface com o usuário. Além disso, a portabilidade aumenta o risco de perda ou danos no dispositivo móvel. Quando em movimento, o equipamento móvel altera sua localização e, possivelmente, seu ponto de contato com a rede. Esta natureza dinâmica insere questões sobre o endereço do nó, a localização do usuário, o acesso a recursos e informações dependentes de localização. A conexão sem fio levanta outros obstáculos: comunicação intermitente, com freqüentes desconexões, bloqueio de sinal e *handoff*; restrita e altamente variável largura de banda; alta latência e taxas de erros.

Embora a tecnologia possa evoluir, muitas destas restrições permanecerão, pois são da natureza do ambiente móvel (IMIELINSKI; VISWANATHAN; BADRINATH, 1997). Neste ambiente, a **adaptabilidade** torna-se o mecanismo essencial para as aplicações (NOBLE, 1998), as quais tentam ajustar algum aspecto da aplicação às alterações no ambiente de execução.

A.1.3 Gerenciamento da Mobilidade Física

Tradicionalmente, o gerenciamento da mobilidade é dividido em:

a) mobilidade pessoal – envolve a identificação do usuário sem requerer qualquer relação fixa entre ele, o terminal e a rede;
b) mobilidade terminal – é o aspecto inerente às redes sem fio, inclui funções para rastrear a localização corrente do dispositivo e manter conexões ao sistema.

Sistemas *pervasivos* requerem o acesso ao sistema em escala global. Incluindo, assim, a necessidade de expandir o gerenciamento para:

a) mobilidade de recursos e serviços – devido à mobilidade do usuário/terminal, os recursos computacionais (dados, código, recursos e serviços) devem estar disponíveis independente da localização;
b) mobilidade global – integra as redes de comunicação futuras e as existentes, incluindo uma grande variedade de protocolos e padrões.

propriedade cenários	Portabilidade	Mobilidade	Conectividade	Ambiente Computacional
Computação nômade	Restrita ao equipamento portátil próprio	Física, associada ao equipamento	Esporádica, associada à rede fixa	Limitado ao dispositivo em uso, e eventual cliente fixo de uma rede. Centrado em aplicações de administração pessoal.
Computação com rede sem fio	Restrita ao equipamento portátil em uso	Física, associada ao usuário e dispositivo, restrita à área de acesso da rede[12].	Intermitente, Restrita à área de acesso da rede	ambiente computacional da rede fixa
Mobilidade da computação		Lógica e global, associada aos elementos dados, código e/ou estado de execução	Permanente, Uso da infra-estrutura da Internet	Ambiente computacional da rede fixa.
Computação Móvel	Restrita ao equipamento portátil próprio	Conexão sem fio, restrita á área de acesso local da rede	Intermitente, Normalmente funcionando em modo desconectado. Redes infra-estruturadas e/ou ad-hoc (ponto-ponto). Amplitude local da rede.	Ambiente computacional da rede fixa, com acesso via rede sem fio. Adaptabilidade para recursos limitados. Centrado no acesso a informações para o usuário móvel.
Computação *pervasiva*	Irrestrita, uso de variados dispositivos móveis	Total (física e lógica, Global) associada à rede *wide area* móvel	Intermitente, acesso à infra-estrutura fixa, e comunicação com computadores na vizinhança (ad-hoc). Amplitude global da rede.	Centrado no usuário e suas atividades. "Segue" o usuário, disponível em qualquer lugar a qualquer tempo, em qualquer dispositivo.

Quadro A.2: Cenários Móveis e suas Propriedades

Figura A.3 ilustra a arquitetura de gerenciamento da mobilidade. A rede de acesso pode incluir redes celulares, redes de comunicação pessoal, redes sem fio, numa dimensão de macro a pico células. A rede *backbone* inclui várias redes de alta velocidade, como FDDI e ATM, que podem estar conectadas à Internet. O gerenciamento da mobilidade, inicialmente desenvolvido para suporte a mobilidade de usuário e terminal, é estendida para gerenciar mobilidade de serviços e recursos.

Esforços para explorar essa nova tecnologia, um misto de redes com e sem fio com aplicações e usuários móveis, estão sendo empreendidos em muitos projetos, como visto ao longo deste texto.

[12] Aplicações recentes abordam agentes móveis para uso em redes sem fio, o quais apresentam a propriedade de mobilidade lógica.

A.1.4 Outros Aspectos Importantes para a Mobilidade

Outro aspecto importante a considerar é o tipo de conexão à rede. Pesquisas, iniciadas no sistema de arquivos móveis Coda (SATYANARAYANAN, 1996; SATYANARAYANAN, 1996b), tem mostrado que existem três modos de operação presentes no ambiente móvel: (i) fortemente conectado – quando usando os recursos da rede fixa/cabeada; (ii) fracamente conectado – quando usando a rede sem fio; (iii) desconectado, sem conexão à rede. O modo desconectado é voluntário e usado para economizar os recursos do dispositivo móvel, como a energia. Assim, considera-se que o equipamento móvel opera neste modo a maior parte do tempo. O tratamento da desconexão/reconexão é um aspecto importante para as redes móveis e para os sistemas e aplicações que operam no ambiente *pervasivo*.

Três outros aspectos, presentes em sistemas distribuídos, adquirem relevância no ambiente *pervasivo*: escalabilidade – inserido pelo tamanho potencial de usuários no sistema; heterogeneidade – inserido pelos diferentes dispositivos e redes presentes no ambiente; dinamismo – inserido pelas alterações na disponibilidade dos recursos e pela mobilidade lógica e física presente no ambiente.

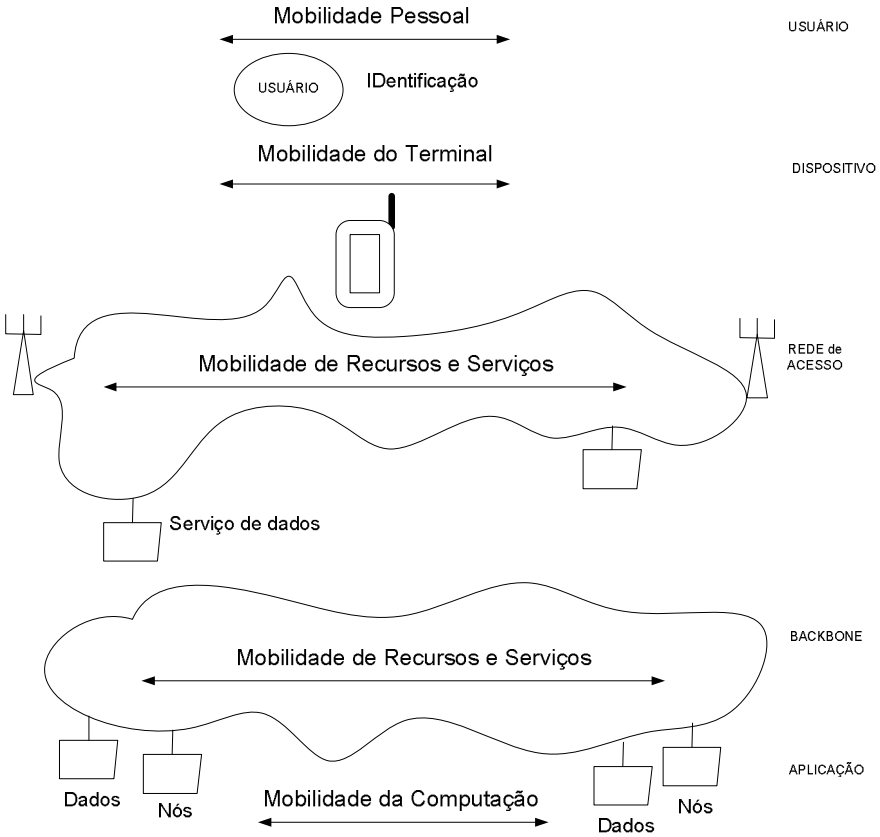


Figura A.3: Arquitetura do Gerenciamento da Mobilidade Física

O projeto de um sistema e aplicações para o ambiente *pervasivo* deve levar em conta estes aspectos. O cenário da Computação *Pervasiva* é o foco da arquitetura ISAM. Para permitir o movimento de todos os elementos (ou combinação destes) ambos, o sistema e a aplicação, devem ter uma funcionalidade do tipo siga-me (*follow-me*). A cena mais ilustrativa deste ambiente móvel é aquela na qual o usuário, carregando seu dispositivo portátil, tais como *palmtops* ou *notebooks*, tem acesso a infra-estrutura compartilhada independentemente de sua localização física ou forma de deslocamento. As aplicações móveis disponíveis hoje, usando redes sem fio local (WLAN) ou da computação

nômade, limitam a liberdade permitida pela mobilidade física. Ainda não se dispõe de uma rede móvel global que permita este deslocamento, mas previsões indicam que esta será realidade em um futuro próximo (SATYANARAYANAN, 2001).

A.2 Aplicações Móveis

Dentre as aplicações móveis, disponíveis atualmente, dominam as **aplicações pessoais** para administração de atividades, como agendas, ou acesso a informações como as *WAP pages* e *e-mails*. Algumas aplicações comerciais são relativas ao acesso a informações em **Banco de Dados Móveis**, as quais replicam um subconjunto de dados para o cliente móvel, e necessitam de um procedimento de sincronização ao final da atividade para a atualização da base de dados primária. Grande parte dos estudos sobre mobilidade, realizados na década de 90, foram sobre o tema acesso a dados. Um estudo relativo a este tema foi realizado pela autora e está disponível na referência (AUGUSTIN, 2000) e na página do projeto ISAM (www.inf.ufrgs.br/~isam).

Embora o acesso a dados seja uma parte importante das aplicações móveis, construir aplicações somente com **mobilidade de dados** é muito restritivo. Acredita-se que as possibilidades oferecidas pela mobilidade lógica e física, a qual projeta uma nova classe de aplicações, devem ser mais exploradas. Para tal, é necessário dispor-se de suporte para o projeto e implementação de aplicações. Este suporte é ainda incipiente, e está em constante evolução. Vive-se, neste momento, um período de transição da tecnologia de Computação Pessoal (nômade) para a Computação Móvel, usando redes sem fio, com possibilidade de acesso permanente à rede.

A.3 Tecnologias para a Computação Móvel

Os principais componentes da Computação Móvel, no nível de *hardware*, envolvem os computadores portáteis, usados pelos usuários móveis, e as redes sem fio, com seus respectivos meios de transmissão de dados e padrões de comunicação de dados. Além disso, considerando que a Internet é o principal meio de comunicação usado hoje e que o uso de sua infra-estrutura é aplicável aos trabalhadores móveis, estão surgindo muitas propostas de adequação da arquitetura TCP/IP à mobilidade, dentre elas o protocolo WAP (*Wireless Application Protocol*). Quanto ao software, tem-se basicamente o conceito de mobilidade de código e de agentes móveis (FUGETTA; PICCO; VIGTNA, 1998; VIGNA, 1998). As principais características e funcionalidades desses elementos são apresentadas em (AUGUSTIN, 2000). A seguir, faz-se um resumo destas tecnologias.

Tecnologias de Hardware

Os dispositivos para o usuário final da Computação Móvel têm várias formas e configurações. As plataformas de *hardware*, o sistema operacional e as capacidades funcionais variam entre esses dispositivos. Entretanto, existem atributos que são compartilhados entre *notebooks*, *palmtops*, computadores *handhelds* e outros acessórios, tais como impressoras, fax e *scanners* móveis. Os **computadores portáteis** devem ser de tamanho reduzido e de pouco peso. Precisam ser resistentes, funcionais, práticos e portáteis. O desenvolvimento de computadores móveis deve conciliar objetivos que são conflitantes: oferecer recursos e desempenho semelhante aos do *desktop* com tamanho, peso e

consumo de energia reduzidos. Também devem oferecer acesso remoto a outros computadores, servidores e *mainframes*, comunicação em rede e suporte a uma crescente variedade de mídias.

Apesar de os *notebooks* serem os pioneiros na Computação Móvel, a popularização de transportar um computador para qualquer lugar veio com o lançamento da primeira versão do Palm, da *Palm Computing* (www.palm.com), chamado de PalmPilot, ocorrido em 1996. Desde então uma crescente evolução na capacidade e funcionalidade destes dispositivos tem sido observada. A Figura A.4 mostra alguns modelos da família Palm: Palm VII, Palm m515, Palm m515 com wireless LAN e Tungsten.

Figura A.4: Alguns Dispositivos da Família *Palm Computing*

Outra linha de dispositivos usados é a de **telefones celulares**. A 2.5ª geração dos celulares, conhecida como *Personal Communciations Services* (PCS) dispõe de serviços que facilitam a comunicação e permitem ao usuário executar certas funções, como enviar mensagens, e o acesso à Internet móvel com o protocolo WAP (*Wireless Application Protocol* - http://www.wapforum.org). A 3ª geração de celulares trafegará a 2Mbps e permitirá aplicações de multimídia e videoconferência, embora ainda não contemple *roaming* global. A Figura A.5 mostra uma pequena coleção de dispositivos móveis.

Figura A.5: Exemplos de Dispositivos Móveis

Este é um mercado que tem apresentado muitas novidades, e a tendência parece ser a convergência entre *palmtops* (computação) e celulares (comunicação) (Interactions, 2002). Como se dará esta convergência é uma equação ainda não resolvida. Alguns aparelhos chegaram ao mercado recentemente, como ilustrado na Figura A.6: o modelo pdQ da Kyocera e o Nokia 9210ig.

Figura A.6: Exemplos de Telefones Inteligentes (Palmtop + celular)

A.3.2 Redes Sem Fio

À medida que cresce o número de dispositivos de comunicação e computadores portáteis, cresce a demanda em conectá-los ao mundo externo. Como não é possível fazer a conexão por fio em todos os ambientes, como a partir de carros e aeronaves, ou em áreas geograficamente acidentadas, as redes sem fio estão se tornando uma opção para tais ambientes. Porém, devido à diversidade dos meios de transmissão e dos protocolos de comunicação, as redes sem fio têm inúmeros formatos e padrões. Atualmente, dois padrões estão dominando o mercado sem fio: Wi-Fi (*wireless fidelity*) ou IEEE 802.11 (LEE, 2001) e Bluetooth (www.bluetooth.org).

O *IEEE Standards Committee* definiu para as redes locais sem fio (WLAN) o padrão 802.11 – *Wireless LAN*, uma extensão do protocolo *Ethernet*. Os primeiros dispositivos dentro deste padrão foram disponibilizados no mercado em 1998. A taxa de transmissão vem evoluindo, dos 2Mbps aos 54Mbps (IEEE 802.11g) atuais.

O protocolo Bluetooth, criado inicialmente pela Ericsson, é definido por um consórcio de empresas de computação e comunicação. É uma opção para comunicação a curta distância entre vários dispositivos, não tendo o inconveniente da comunicação por infravermelho que necessita de "estar na vista" para estabelecer a conexão. A comunicação é por rádio, par-a-par.

Segundo o grupo de estudo 802.11 do IEEE (*Institute of Electrical and Electronic Engineers*), as redes sem fio podem ser classificadas em redes infra-estruturadas e redes independentes ou *ad-hoc*.. As **redes infra-estruturadas** consistem basicamente de uma parte fixa e de uma parte móvel (Figura A.7).

A parte móvel contém dispositivos móveis, como *palmtops* ou *notebooks*, capazes de se comunicarem com a parte fixa ou com outros dispositivos móveis. Esses computadores, também chamados de unidades móveis (UMs), não possuem uma posição fixa na rede, podendo se mover ao longo da rede, mesmo durante uma comunicação. As UMs comunicam-se com as estações-base através de conexões sem fio. Cada unidade móvel possui uma estação-base que funciona como seu ponto de referência, sua *home*. A parte fixa corresponde às redes e computadores tradicionais com características adicionais, tais como a capacidade de comunicação com dispositivos

sem fio e funções de gerenciamento da mobilidade. Os computadores da parte fixa são conhecidos como estações rádio-base (ERB) ou estações-base. A área de abrangência de uma estação-base é chamada de célula. Para suportar taxas maiores de transmissão, o diâmetro das células geralmente é da ordem de 100m para ambientes externos (microcélulas), como um campus, ou da ordem de 10m para ambientes internos (picocélulas). Quanto menor o tamanho da célula, mais freqüente é a ocorrência de *handoffs* (mudanças fronteiriças entre as células).

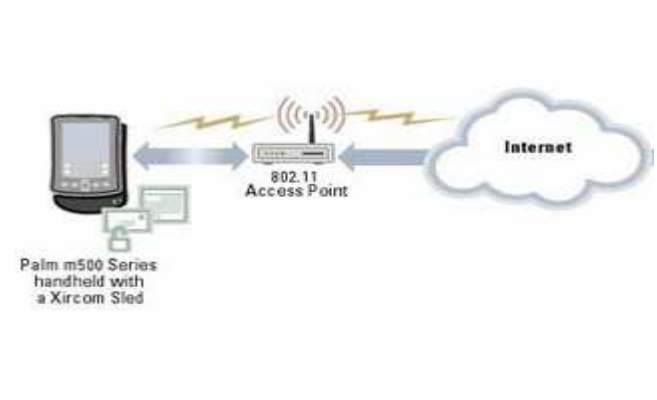


Figura A.7: Redes Infra-estruturada com Wi-Fi

O outro tipo de rede móvel é a **rede *ad-hoc*** (MANET – *Mobile Ad Hoc NETwork*), também conhecida como *Mobile Packet Radio Networking*, onde os dispositivos móveis são capazes de trocar informações diretamente entre si, sem passar por uma central (Figura A.8). A rede *ad-hoc* suporta "*plug-ins*" temporários de conexões sem fio de vários nodos. Essas redes devem oferecer facilidades de conexão à rede e de acesso aos serviços oferecidos pela rede. Um dos problemas fundamentais dessas redes é manter as rotas, já que a mobilidade de um dispositivo pode causar mudanças na topologia. O padrão Bluetooth está sendo usado para construir este tipo de rede.

Figura A.8: Redes Ad-hoc com Bluetooth

Algumas das vantagens das redes *ad hoc* são: a rápida instalação em locais sem infra-estrutura prévia, a reconfiguração dinâmica da rede, a flexibilidade da conectividade e a mobilidade. Como desvantagens têm-se: menor banda passante, taxa

de erros no enlace sem fio, problema em determinar a localização física do dispositivo móvel, e dificuldade no roteamento devido ao movimento não-determinístico dos dispositivos móveis.

A.3.3 Tecnologias de Software

Os dispositivos móveis atuais têm diferentes formas e tamanhos, e objetivam diferentes mercados tendo diferentes requisitos. Considerando o mercado de dispositivos para acesso sem fio de informações há uma variedade de dispositivos desde os tradicionais PDAs (*Personal Digital Assistants*), com funcionalidade de computador e comunicação ocasional, até telefones celulares, com comunicação de voz e capacidade para suporte a livros de endereço e mensagens básicas. Logo, desenvolver aplicações nestas plataformas exige conhecer a forma de operação do seu sistema operacional, e as ferramentas de desenvolvimento disponibilizadas. As duas mais importantes tecnologias para a produção de software com mobilidade são: a linguagem Java e agentes móveis.

A.3.3.1 Sistemas Operacionais e Ferramentas de Desenvolvimento dos PDAs

Os sistemas operacionais dos PDAs têm requisitos resultantes da necessidade de tratar as restrições e capacidades limitadas do hardware. As características-chaves dos sistemas operacionais de PDAs são: (a) rápida carga de processos; (b) resposta imediata ao chaveamento de uma aplicação para outra; (c) eficiente operação de sincronização/back-up. Os principais sistemas operacionais são WindowsCE, PalmOs, SymbianOS e Linux.

WindowsCE

A última versão do windowsCE, conhecida como PocketPC, tem uma parceria da Microsoft com fabricantes de equipamentos **PocketPC**: Casio, HP e Compaq (www.microsoft.com/windows/embedded). O típico windowsCE é um sistema embutido que objetiva atender um usuário específico, executa desconectado de outros computadores e requer tratamento eficiente de interrupções. O kernel do SO é responsável pelo escalonamento e sincronização de *threads*, processamento de exceções e interrupções, carga das aplicações e gerenciamento da memória virtual.

WindowsCE.NET é o nome corrente para a próxima versão do windowsCE que objetiva a próxima geração de dispositivos baseados em comunicação sem fio e conexão à infra-estrutura existente.

WindowsCE oferece interfaces de programação para o ambiente Microsoft Win32, controles ActiveX, interfaces COM (*Component Object Model*), suporte para multimídia (DirectX), suporte a comunicação (TCP/IP, SNMP, e outros), segurança, e outros. Uma variedade de aplicações integradas, como Pocket Internet Explorer, Pocket Outlook, Pocket Word, expõem objetos que podem ser customizados ou estendidos para criar novas aplicações.

PalmOS

O PalmOS é o sistema operacional da família de dispositivos *palms*, da Palm Computing – subdivisão da 3Com, e de dispositivos de outros fabricantes como Sony, IBM e Qualcomm (www.palmos.com). O SO é multitarefa, mas somente uma tarefa é para aplicações, ou seja, uma aplicação deverá terminar antes que outra possa ser executada. O espaço do sistema necessário para qualquer aplicação que está executando é a memória RAM, dinâmica e reusável. Aplicações relativas a bases de dados (.pdb)

são armazenadas em meio, chamado permanente, que também usa a memória RAM (não reusável, neste caso). A aplicação é dividida em código executável (.prc) e diferentes elementos como interface e ícones, de forma de estes possam ser alterados sem reescrita do código.

A próxima versão do PalmOS, PalmOS 6 prevista para o final de 2003, deverá ser uma simbiose entre o PalmOS atual e BeOS, com vistas a atender o mercado de ferramentas ubíquas. A versão atual, PalmOs 5.2, focalizou melhorias no tratamento de dados multimídia e no acesso sem fio seguro. Tem suporte à comunicação usando diversos mecanismos (http, socket, Bluetooth). Porém, ainda mantém a restrição de uma única aplicação de usuário executando por vez no dispositivo, e a limitação de 64k para o tamanho desta aplicação.

O PalmOS *Software Development Kit* (SDK) fornece interfaces de programação (APIs) para o desenvolvimento de aplicações, com funções para interfaces do usuário, gerenciamento do sistema e de comunicações. As linguagens mais usadas são versões compactas de C e Java. O *Conduit Development Kit* (CDK) suporta a implementação de sincronizadores de dados entre a aplicação no *desktop* (Windows, Mac e Solaris) e a aplicação executando no dispositivo. Além disso, kits e ferramentas para vários ambientes de desenvolvimento, como Kit para Java *wireless* (www.sun.com/products/midp4palm), dão suporte a programação de novas aplicações.

SymbianOS

SymbianOS é o sistema operacional projetado para pequenos dispositivos baseados em telefones celulares, originário do primeiro sistema do Psion (www.psion.com), que servem como "dispositivos de informações sem fio" (www.symbian.com). O código do SO é compacto e armazenado em um chip ROM. Embora possa ser portado para outros microprocessadores, Symbian é baseado na arquitetura ARM (*advanced RISC Machine*). O SymbianOS inclui um *kernel* multitarefa, *middleware* para comunicação, gerenciamento de dados e gráficos, *GUI framework* de baixo nível e máquina de execução de aplicações. Implementa a maioria das multitarefas através de mensagens dirigidas por eventos, mais que *multithreading*, como outros sistemas operacionais. A força desta plataforma é a comunicação: e-mail Internet usando POP3, IMAP4, SMTP, MHTML, e troca de mensagens usando SMS; protocolos de telefonia móvel; protocolos de comunicação TCP/IP, Bluetooth, WAP, IrDA, serial.

Symbian fornece kits de desenvolvimento para C++ e Java. Programadores escrevem o código no PC e usam emuladores para testá-lo.

Linux

Linux em PDAs, também chamado *Embedded Linux Edition*, tem seu maior representante, atualmente, no Zaurus SL-Sxxx da Sharp (www.zaurus.com), que roda o *Embedix Plus PDA* da Líneo Inc., baseado no kernel do Linux 2.4 (www.lineo.com/products/embedded). O Embedix possui utilitários para tratamento de *files, shell, find, text, grep, tar* e outros, conforme implementação-base do BusyBox (www.busybox.net) – projeto de código aberto mantido pela Lineo. A interface gráfica do Zaurus é Qt/Embedded, similar ao KDE, da Trolltech, a qual inclui aplicações PIM, clientes Internet, jogos e utilitários. O navegador é o Opera. Outra plataforma de software importante do Zaurus é a *Jeode PDA Edition*, da Insígnia Solutions (www.insignia.com), que implementa Java e o perfil *Personal Profile* (ver Seção A.2.3.2).

Outro equipamento que tem versão Linux é o iPAQ da Compaq/HP, com o processador StrongARM. Enquanto a Compaq focaliza no kernel Linux, outros projetos

estão em andamento para fornecer novas distribuições. O Familiar GNU/Linux é a distribuição de referência (http://familiar.handhelds.org). Esta contém os pedaços necessários para carregar o Linux, módulos para interface com dispositivos de entrada, rede, display, etc..., e o gerente de pacotes (ipkg), baseado nos pacotes Debian para *desktop Linux*. As informações na página http://mstempin.ree.fr/linux_ipaq ensinam a instalar o Linux no iPAQ com a interface gráfica Opie. Outro projeto de distribuição, Intimate, é uma extensão do Familiar que oferece mais ferramentas, porém exige maiores recursos do PDA (http://intimate.handhelds.org). Projetos de implementações específicas que portam o Linux do ambiente desktops para PDAs também estão em expansão, como Qt Palmtop Environment, da Trolltech, para o iPAQ (www.trolltech.com/ products/qt/embedded).

A.3.3.2 Linguagem Java

A versão *small-Java* (www.sun.com), da Sun, lançada em 1999, fornece suporte para programação em equipamentos portáteis com poucos recursos de memória, processamento e armazenamento. Alguns pesquisadores consideram que Java será a plataforma dominante no setor de *wireless* (COMP; DOBBING, 2003). A Figura A.9 apresenta a plataforma Java vigente e destaca os tipos de máquinas as quais se destina.

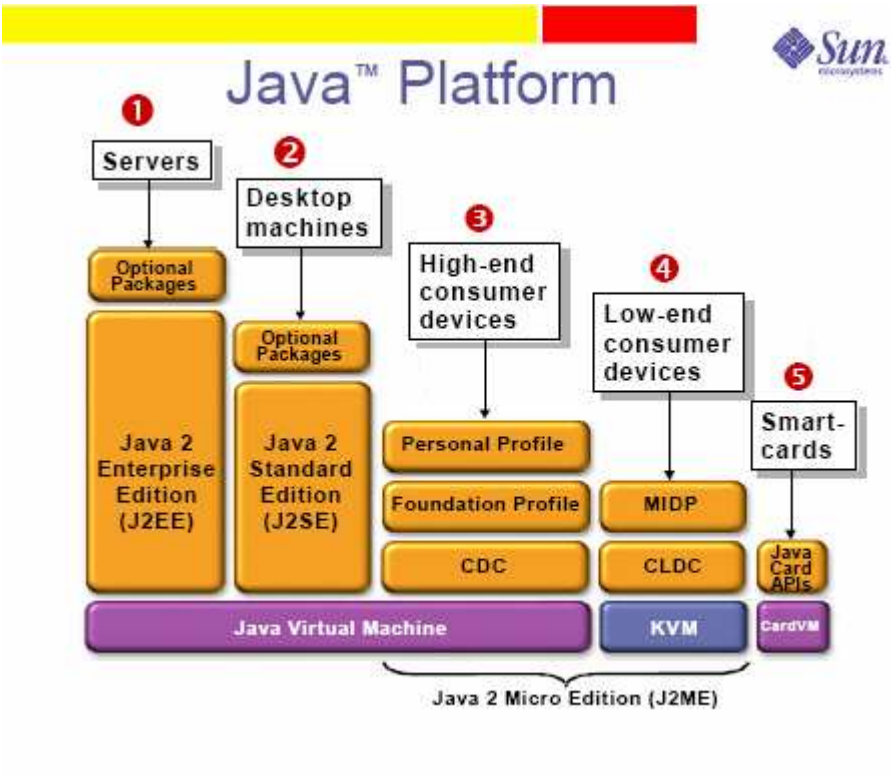


Figura A.9: A Plataforma Java da Sun

J2ME (***Java Micro Edition*** - www.sun.com/j2me) adapta a existente tecnologia Java para computadores de mão (*handhelds*) e dispositivos embutidos, definindo novas classes para tratar com as restrições e limitações desses dispositivos. É uma versão reduzida da linguagem que mantém compatibilidade com esta (Sun Microsystems, 2002). Porém, é importante salientar que esta tecnologia está em um estado de transição, e mudanças de rumo ocorrem constantemente.

KVM e CVM são versões da máquina virtual Java para operar em equipamentos portáteis. Recentemente, a Sun disponibilizou um conjunto de ferramentas para o projeto de aplicações com comunicação sem fio, *Wireless Toolkit Java* (www.sun.com/products/j2MEwtoolkit).

J2ME é estruturado em uma arquitetura modular, e é baseada em dois novos conceitos: configuração (*configuration*) e perfil (*profile*), conforme mostra a Figura A.10. Uma **configuração** define o ambiente de execução básico (*runtime j2me*), incluindo a máquina virtual e novas classes derivadas do J2SE. Cada configuração engloba uma família de dispositivos portáteis com capacidades similares. Até o momento, duas grandes configurações foram definidas: CDC (*connected device configuration*) e CLDC (*connected, limited device configuration*).

A configuração CLDC adapta Java para dispositivos de poucos recursos, baixa capacidade e limitada conectividade, e disponibiliza uma máquina virtual, como a KVM[13], que remove muitas capacidades existentes no *kernel* J2SE. A linha Palm e celulares estão nesta categoria. As limitações do perfil CLDC impostas à máquina virtual Java são, entre outras: não suporte a carga dinâmica de classes, não suporte a reflexão e serialização, não suporte a grupo de *threads* nem *daemon thread*, não suporte a RMI e outras características avançadas de Java. CLDC 1.0 foi liberada em 2000 quando dispositivos móveis tinham baixa capacidade de processamento e memória. Esta implementa os pacotes J2SE: `java.lang`, `java.util`, `java.io`, e introduz `javax.microedition`.

Já, a configuração CDC objetiva dispositivos de maior capacidades, como Zaurus da Sharp (www.zaurus.com), e disponibiliza máquina virtual, como a CVM (www.sun.com/products/cdc/cvm), com as capacidades do *kernel* J2SE. A configuração de máquina CDC prevê 512k de memória ROM, no mínimo; 256k de memória ROM, no mínimo; conectividade à rede; suporte a especificação da JVM; restrições são relativas à interface do usuário.

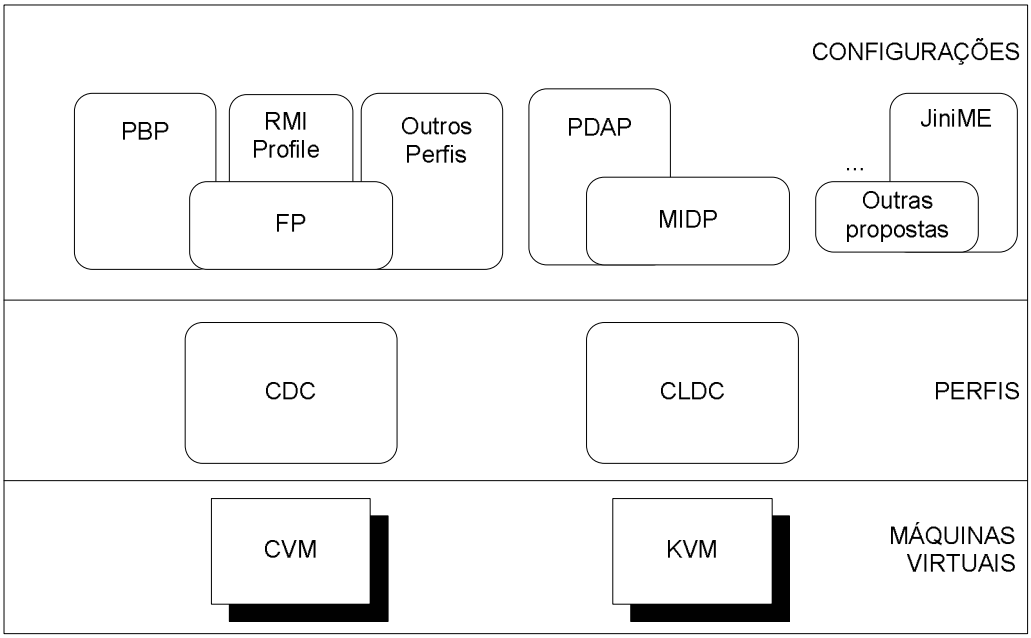


Figura A.10: Arquitetura J2ME

[13] PalmOS é a plataforma de referência escolhida pela Sun para implementar a K Virtual Machine.

Ambas configurações têm classes comuns ao J2SE (*Standard Edition*), de forma a permitir a compatibilidade entre eles (Figura A.11), e classes específicas aos tipos de dispositivos.

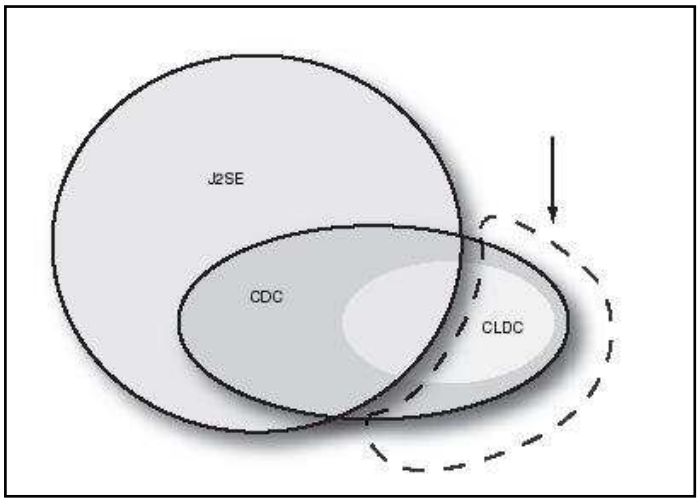


Figura A.11: Relação J2ME e J2SE

Um perfil (*profile*) estende a configuração adicionando classes de domínio específico (uso específico de um dispositivo e ausente na configuração) ao conjunto de classes da configuração. O perfil refere-se a aspectos como interface do usuário, conexões de rede, segurança, persistência, etc. A configuração CLDC inclui o perfil MIDP (*mobile interface device profile*) e o perfil PDAP (*personal digital assistant profile*). A configuração CDC inclui os perfils: FP (*foundation profile*), PBP (*personal basis profile*) e PP (*personal profile*).

MIDP foi o primeiro perfil implementado e é o mais maduro (Sun Microsystems, 2002). A versão 1.0 implementa APIS para rede baseadas no *Generic Connection Framework* (GCF) - conexão via http, datagrama ou *streams*; interface de usuário; persistência local baseada em banco de dados orientados a registros; e ciclo de vida das aplicações MIDlet. A versão 2.0 adiciona APIs para rede, incluindo sockets (TCP), datagramas UDP, serial, push, e conexões seguras (https, certificação); sons e jogos.

PDAP é o primeiro perfil específico para PDAs, pois no MIDP faltam especificações para gerenciamento de informações pessoais, como livros de endereços, agendas e outros. Também, PDAs tendem a ter capacidades maiores de armazenamento e memória que celulares, e necessitam acesso à porta serial, aos arquivos e uma interface mais rica. PDAP estende MIDP para essas funcionalidades. A especificação 1.0 deste perfil já foi aprovada (JSR 75), mas ainda não tem implementação disponível para *download*. A especificação prevê o pacote PIM (*Personal Information Management*) para acesso ao gerenciamento de dados nativos (`javax.microedition.pim`); pacotes de interface como AWT (`java.awt.*`); pacote de suporte à rede baseada no GCF/MIDP (`javax.microedition.io`); e pacotes adicionais correspondento ao J2SE, como `java.io, java.lang.reflect, java.net, java.util`.

O perfil FP estende CDC com mais classes J2SE de suporte à segurança e utilidades para programação. Atua como base para a construção de outros perfis. Porém, não fornece nenhuma classe de interface de usuário (`awt`). Este é usado para programar dispositivos sem interface com usuário, dispositivos embutidos e dedicados. O perfil PBP fornece um conjunto mínimo de classes para programação de interface com usuário, suporte a `JavaBeans`, e modelo de aplicação `Xlet`.

O perfil PP contém os anteriores e adiciona classes AWT e suporte a applets. É o ambiente mais completo e similar ao *kernel* do J2SE, e requer dispositivos de no mínimo 512Mb de memória. Basicamente, especifica a implementação dos pacotes: `java.lang`, `java.util`, `java.net`, `java.io`, `java.text` e `java.security`. A plataforma Java da Insígnia Solution, Jeode, para o Zaurus da Sharp, implementa o perfil *Personal Profile* 1.2 (PP) da configuração CDC (www.sun.com/products/personalprofile), a qual permite a este dispositivo executar aplicações compatíveis com J2SE padrão (http://developer.java.sun.com/developer/earlyAccess/pp4Zaurus). O ambiente de desenvolvimento para a série Zaurus SL-series é o J2SE 1.4.1. A implementação evoluiu do *Personal Java Runtime Environment* (pJava) e inclui uma JVM similar a do J2SE. A ferramenta *Personal Java Emulation Environment* (PJEE) simula o dispositivo no desktop, e é usada para testes de aplicações. Outra ferramenta disponível é o JavaCheck, que faz análise dos arquivos *.class. O Zaurus vem com o *Personal Java Runtime Environment* (EVM) instalado, máquina virtual similar a CVM.

O *toolkit J2ME da SUN* também fornece pacotes adicionais: APIs para Bluetooth, Wireless Messaging API, Mobile Media API e RMI, para dispositivos com suporte a CDC/FP.

As aplicações são desenvolvidas usando essas ferramentas, e testadas com o auxílio de emuladores/simuladores de dispositivos móveis, como celulares e palmtops – desenvolvimento multiplataforma. Os emulares/simuladores são fornecidos pelos fabricantes. Por exemplo, a Palm fornece emuladores para PalmOS 3.5 (POSE), e simuladores para PalmOS 5.0 em sua página (www.palmos.com/dev/tools). A aplicação é transferida para o dispositivo usando ferramenta fornecida pelo fabricante. Por exemplo, *HotSync Manager Program, ou Jsync* da Palm (www.palmos.com/dev/techn/conduits) para trocar e sincronizar dados com o desktop.

ANÁLISE

Diferente da indústria de tecnologia da informação e da computação pessoal, em sua história recente, o mercado móvel tem experimentado uma significativa fragmentação nos ambientes de desenvolvimento de aplicações. Isto se evidencia nas especificações de configurações, que atendem as necessidades horizontais do mercado, e nos perfis, que atendem as necessidades verticais do mercado sem fio. Esta fragmentação tem origem no fato de que as especificações foram feitas em uma época, por volta de 1999, em que os dispositivos tinham quantidade de memória e de poder de processamento limitada. Avanços recentes na tecnologia de hardware têm eliminado muitas restrições, e torna-se comum encontrar dispositivos com velocidade de processamento de 200-400MHz e memória de 40Mb, muito além dos 328Kb de memória da especificação CLDC/MIDP. A constante remoção dessas limitações dirige as aplicações para o perfil CDC/PP, que é similar a plataforma J2SE 1.2, fazendo com que estas convirjam para o uso reunificado de uma única tecnologia Java: J2ME integrada com J2EE e J2SE. Esta reunificação deverá ocorrer num futuro próximo.

A.3.3.3 Mobilidade de Código

Segundo Cardelli (1998), existem duas noções distintas de mobilidade: mobilidade da computação (*mobile computation*), que enfatiza somente a mobilidade do *software*, e a Computação Móvel (*Mobile Computing*), que enfatiza também a mobilidade física. Esses dois aspectos hoje estão separados, porém, os limites entre mobilidade virtual e

física são imprecisos, e eventualmente terá que se tratar todos os tipos de mobilidade de forma uniforme.

Informalmente, pode-se definir **mobilidade de código** como a capacidade de dinamicamente alterar a ligação entre o fragmento de código e a localização onde este executa. Vigna (VIGNA, 1998), em sua tese, criou uma taxonomia das linguagens para código móvel dividindo-as basicamente em duas correntes: **mobilidade forte**, na qual a unidade de mobilidade compreende o código e o estado de execução; e **mobilidade fraca**, onde somente o código é transferido. Nesta classificação, Java e Obliq (CARDELLI, 1995) são linguagens com mobilidade fraca, enquanto Agent Tcl (GRAy, 1996) e Telescript (WHITE, 1996) são linguagens com mobilidade forte.

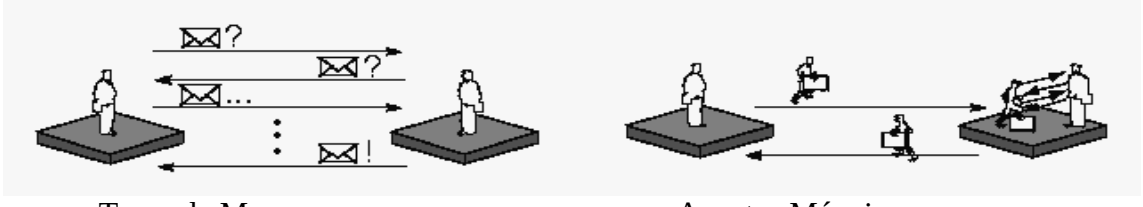
Vigna (1998) analisou várias linguagens e concluiu que três abordagens são utilizadas para construir aplicações com código móvel:

a) avaliação remota, onde um componente invoca serviços em outro componente, enviando a este não somente os dados mas também o código de como executar o serviço;
b) agentes móveis, onde componentes viajam pela rede para locais que tem os recursos de que necessitam, levando seu código e estado interno;
c) código sob demanda, onde o código é carregado de um "servidor de código" para a execução de uma tarefa. Um exemplo são os *applets* de Java.

Em sua tese, Vigna analisa as aplicações mais apropriadas a cada uma das abordagens acima, e sugere que as duas primeiras são apropriadas para as aplicações da Computação Móvel. Como as unidades móveis são fracamente acopladas, a troca de mensagens remotas é inadequada à interação com computadores móveis. **Agentes móveis**, no entanto, eliminam a necessidade de permanência da conexão. Os agentes executam o código da iteração assincronamente e localmente ao nó destino, em oposição ao diálogo remoto e síncrono da troca de mensagens. Assim, não requerem que o dispositivo original esteja disponível durante a interação. De fato, agentes móveis tiram vantagem dos recursos estacionários em nome da unidade móvel movendo dinamicamente porções críticas da aplicação para nós fixos da rede. A Figura A.12 ilustra essa diferença.

O projeto D´Agent, da Dartmouth College, é um dos projetos que procuram solucionar as questões envolvendo agentes no ambiente da Computação Móvel. Segundo Gray (1997), um dos líderes do projeto, no ambiente da Computação Móvel, os sistemas de agentes devem suportar operações que capacitam o agente a:

a) mover-se de um computador parcialmente conectado (tais como *laptop*) e retornar mais tarde, ainda que o computador esteja conectado somente por breves períodos e altere sua localização na reconexão;
b) navegar através da rede para encontrar os serviços dos quais necessita;
c) "sentir" e reagir ao ambiente de rede, atuando de forma autônoma enquanto o usuário está desconectado;
d) comunicar-se efetivamente com outros agentes.



Troca de Mensagens Agentes Móveis

Figura A.12: Modelos de Interação Remota

Além da redução da dependência da conexão, os agentes móveis também fornecem soluções para problemas causados pela mobilidade do dispositivo em si. O agente permanecendo na parte estacionária da rede pode atuar como substituto/em nome do usuário móvel em aplicações onde este não é alcançável ou não está conectado. Outra vantagem vem da economia no custo da transmissão e da habilidade de o agente dinamicamente mover seu local de execução. Cada domínio pode oferecer serviços locais para os computadores que entram em seu raio de ação. Isto ilustra o potencial deste modelo para alcançar um tipo de configuração remota e dinâmica.

O maior interesse da Computação Móvel nesta solução é o potencial de que "o agente se move para onde os recursos estão". Porém, em geral, atualmente os agentes se movem explicitamente entre localizações. Poucos sistemas incorporam a descoberta e gerenciamento dos recursos para orientar a decisão de para qual localização o agente deve se mover. Dentre estes, citam-se a linguagem Sumatra (RANGANATHAN; ACHARYA; SALTZ, 1997), que estende Java para o gerenciamento de recursos, e o sistema de agentes móveis Ara (PEINE; STOLPMANN, 1997). Nesses casos, as aplicações devem usar informações *online* sobre o ambiente operacional e conhecer suas próprias necessidades em recursos para tomar decisões sobre a localização da computação e dos dados. Soluções para o gerenciamento e descoberta de recursos, bem como dos mecanismos para adaptação às variabilidades dos recursos permanecem um ponto aberto de pesquisa.

Plataformas de agentes móveis para PDAs é outro ponto de pesquisa em aberto. Considerando as limitações impostas pelo perfil CLDC, construir agentes móveis em J2ME não é possível sem a disponibilidade de uma plataforma de suporte que permita implementar mobilidade de objetos (carga dinâmica de classes, serialização e reflexão). Na *Monash University* (MIHAILESCU; KENDALL, 2001) está se desenvolvendo projetos que implementam uma plataforma de agentes para PDAs. Estes usam J2ME e propõe incluir a plataforma de agentes na KVM. Dragomirescu (2000) propõe um *ClassLoader* dinâmico pra KVM em PalmOS. JiniME (*Jini Micro Edition*) é a proposta de uma configuração sobre CLDC para Jini (KAMINSKI, 2000) que objetiva fornecer: serialização de objetos, download de base de código, carga dinâmica de classes, invocação remota de serviços, serviços de *lookup*, e protocolos de descoberta de *bootstrapping*. Estas pesquisas indicam que futuras versões do CLDC terão suporte para a mobilidade de objetos, descoberta de recursos e serviços, e correspondentes melhorias na Máquina Virtual Java, com vistas a atingir melhor desempenho.

APÊNDICE B RECONHECIMENTO DO CONTEXTO

Hoje, o contexto é difícil de usar porque, diferente de outras formas de entradas de dados, não existe uma forma comum, reusável de tratá-lo (DEY, 2000). As aplicações conscientes do contexto têm sido implementadas de forma *ad-hoc*, fortemente influenciadas pela tecnologia usada para adquirir dados dos sensores. Em geral, adquirir informações de contexto é parte integrante da aplicação, tornando as soluções muito específicas para uma aplicação ou domínio de aplicação, como as aplicações multimídias (ANGIN et al., 1998; NOBLE, 2000). Isto dificulta o reuso da solução e a própria evolução da aplicação.

A generalidade da solução ISAMadapt requer um suporte para aquisição e notificação dos mais diversos elementos de contexto. Estudos visando determinar o estado-da-arte da *Context-aware Computing* revelaram que não existe, disponível, um sistema que atenda os requisitos da mobilidade ISAM. Assim, optou-se por projetar e implementar um protótipo do serviço de reconhecimento do contexto que dá suporte à execução de aplicações móveis de propósito-geral e conscientes do contexto. Este serviço é fornecido através do serviço de reconhecimento de contexto, chamado `ISAMcontextService`, descrito neste Capítulo. A implementação desse middleware é o foco do **projeto contextS**, financiado pelo CNPq/SEPIN/FINEP no edital PDI&TI/2002.

B.1 Requisitos de Projeto

Os requisitos gerais do projeto do servidor de reconhecimento de contexto são independência, transparência, evolução e escalabilidade.

Independência

`ISAMcontextService` é parte integrante do ISAM e executa independente das aplicações. Esta independência permite separar as questões relativas a como as informações de contexto são adquiridas de como são usadas. Também, permite o reuso da mesma informação por várias aplicações, e a sobrevida das informações de contexto depois que a aplicação terminou. A independência do serviço em relação às aplicações é mantida através do modelo de relacionamento *publish-subscribe* para os elementos de contexto de interesse de cada aplicação. A notificação tem natureza reativa, ou seja, a alteração no estado gera um evento que dispara uma ação no destino. Outro aspecto para manter a independência é o baixo acoplamento entre os componentes do servidor, que usam o formato XML para comunicarem-se entre si, e trocam dados no formato tupla-objeto (ver Seção B.2).

Transparência

O serviço de reconhecimento de contexto objetiva tornar transparente esta parte integrante da aplicação. O projetista da aplicação não necessita se preocupar como o contexto é adquirido, mas somente definir qual é o contexto de interesse e como este será interpretado e usado. Com isto, implementa-se uma forma declarativa de programação. O processo é automático e colaborativo, sendo a aplicação responsável por informar somente os parâmetros requeridos para seu funcionamento, de forma a torná-lo mais afinado com ela. Como dito, o relacionamento entre o servidor de contexto e a aplicação é baseado no paradigma *publish-subscribe*: os entes da aplicação (i) inscrevem-se em um ou mais elementos de contexto, declarando seu interesse em receber notificação de alterações no estado desses elementos, (ii) definem a visão particular de interpretação do estado do contexto. Os elementos são monitorados e alterações são publicadas.

Evolução

Os serviços de tratamento dos dados e da aquisição de dados são módulos plugáveis que podem ser adicionados, alterados ou removidos do sistema, sem alterar a funcionalidade da aplicação. Desta forma, existem algoritmos nativos para os serviços que foram disponibilizados pelo sistema, e algoritmos projetados e registrados no sistema pelos programadores, tornando-os, a partir do registro, integrantes do catálogo do sistema, e outros, específicos de uma aplicação. Estes últimos não são registrados, mas somente instalados, e serão desinstalados quando a aplicação terminar sua execução.

Escalabilidade

A escala também deve ser levada em conta devido à mobilidade global. A infra-estrutura necessita trabalhar com um grande número de sensores, informações coletadas e manipuladas, e aplicações. Para tratar desta questão, a organização do servidor é distribuída entre as `EXEHDAbases`, e gerenciada pelo EXEHDA. Quando um ente da aplicação necessita de uma informação de contexto, este é registrado no servidor da `EXEHDAbase` local; quando o ente migra para outra `EXEHDAbase`, seu registro é transferido, e o servidor da `EXEHDAbase` corrente passa a atendê-lo. Cada célula tem um servidor local responsável por gerenciar os elementos de contexto de interesse dos entes que estão naquela célula. Esta organização é transparente para a aplicação, que sempre se relaciona com uma variável de ambiente, `ISAMcontextService`, que indica o serviço local que atende o ente naquele momento. Esta variável é gerenciada pelo EXEHDA.

Desta forma, os requisitos operacionais para o projeto do serviço são:

a) a organização distribuída par-a-par entre as células;
b) a informação de contexto pode vir de diversas fontes, físicas e lógicas, geograficamente distribuídas;
c) a separação do tratamento dos dados (dados de contexto) da aquisição de dados (dados de sensores);
d) a interpretação dos dados em informações de alto nível requeridas pela aplicação;
e) o tratamento dos dados adquiridos é realizado de diversas formas, entre elas, estatísticas e preditivas;
f) a entrega de informações de forma síncrona, através de consultas, e assíncrona, através de notificações;

g) o gerenciamento da mobilidade dos componentes da aplicação (entes) relativa ao contexto;
h) o uso de tecnologias existentes de sensoramento;
i) a preservação do histórico dos dados;
j) a constante disponibilidade de informações;
k) a comunicação assíncrona entre os componentes do servidor.

Por definição, a aplicação ISAMadapt requer receber o contexto que lhe é útil e na forma de informações de alto nível, conforme sua especificação. O Serviço deve, então, suportar múltiplas especificações de diversas aplicações, que podem requisitar o mesmo dado ou diferentes conjuntos de contexto.

Considerando a engenharia de software, os requisitos buscados são flexibilidade, extensibilidade e generalidade.

Flexibilidade

Alcançada através da determinação dos recursos a serem monitorados pela aplicação e pela parametrização dos componentes do servidor. Estudos indicam que o desempenho das aplicações é governado por poucos recursos (ANDIN et al., 1998). Por exemplo: rede – *throughput*, *delay*, *jitter* e perda; nodo – escalonamento da CPU e alocação da memória; nodo móvel – bateria, memória e armazenamento. Baseado nesta premissa, as aplicações não devem ter mais de dois ou três recursos aos quais são sensíveis e se adaptam às suas variações de estado. Entre estes é estabelecida uma ordem de prioridade para auxiliar a resolução de conflitos, quando alterações ocorrerem ao mesmo tempo nos recursos.

Extensibilidade

Garantida pela organização independente dos componentes do serviço, e pela padronização do modo de obter e representar a informação. O formato dos dados e os protocolos usados para a interface da infra-estrutura de gerenciamento são a “cola” que torna os pedaços da infra-estrutura interoperáveis. Por esta razão, eles necessitam ser simples o bastante para serem implementados em praticamente qualquer dispositivo e serem usados por qualquer aplicação.

Generalidade

Fornecida através da identificação de funcionalidades requeridas para disponibilizar o serviço de reconhecimento do contexto para as aplicações. Estas devem ser parametrizáveis para se adequar às necessidades específicas de cada aplicação.

Para projetar o Serviço, as principais decisões foram relativas ao formato de dados, serviços oferecidos, e aspectos que influenciam a efetividade da adaptação. Estas são comentadas a seguir.

B.2 Formato dos Dados

Além da simplicidade, o formato de dados deve ser rico o bastante para cobrir uma faixa de sensores e tipos de contexto requeridos. As informações de contexto devem ser mnemônicas, assíncronas e escaláveis, acessíveis para qualquer entidade móvel distribuída. O assincronismo é apropriado para desconexão da rede ou alta latência, características do ambiente móvel. O modelo *subscribe-publish* fornece o assincronismo entre o fornecedor do serviço e a aplicação. O modelo de espaço de objetos, com o formato de tuplas-objeto, fornece o desacoplamento temporal e

espacial, além da possibilidade de uso de mnemônicos, e da troca de informações assíncronas e anônimas.

Desta forma, o formato de dados adotado é o de tuplas-objeto armazenadas em um Espaço de Objetos. O protocolo é o gabarito da tupla. Esta pode ter vários formatos, dependendo da necessidade de informação do elemento de contexto sendo monitorado. Em geral, a tupla terá o gabarito `<tagSENSOR, tagCONTEXTelement, VALUE>`. Os dados de contexto podem não estar acessíveis ou disponíveis. Por isto, `UNKNOWN` é um valor válido para todos os tipos de elementos de contexto.

B.3 Serviços Fornecidos

Depois do formato dos dados, o segundo aspecto do projeto foi definir a infra-estrutura de contexto que fornece os vários serviços para a aplicação. Os serviços que integram a infra-estrutura do `ISAMcontextService` são básicos. Serviços especializados, específicos de uma aplicação, devem ser implementados caso-a-caso. Os serviços fornecidos são detalhados a seguir.

B.3.1 Informação Sensorada

O primeiro dos serviços é obter informações do ambiente. Diversos elementos de contexto podem ter várias formas de obter seu estado. Muitos estudos sobre sensoramento já foram realizados, e algoritmos que os implementam estão disponíveis (DEY, 2000). Esta diversidade de técnicas para obter os dados impede uma modelagem genérica de sensores que englobe todas os requisitos peculiares destes. Desta forma, a solução adotada foi colocar o sensoramento como um processo externo à arquitetura de reconhecimento do contexto, de forma a manter a independência das soluções de sensoramento em relação à observação e obtenção de dados do ambiente.

Como os sensores são externos, o `ISAMcontextService` deve registrar os sensores, ativá-los e coletar os dados enviados por estes. O primeiro serviço é provido pelo **registro**, ativação e desativação de componentes (componente `register`), e os demais pelos **módulos de coleta** (componentes `monitor` e `collector`). Os monitores executam em computadores que tem os sensores conectados a eles, e disponibilizam as informações para o coletor. O coletor reúne informações de vários monitores e as disponibiliza para os demais componentes.

Um sensor torna-se disponível no sistema, quando se registra no monitor. O registro compreende informar, entre outros, a localização e a rotina de ativação que deverá ser executada quando o sistema desejar que o sensor comece a operar. O projetista deve conhecer os tipos de dados fornecidos pelo sensor, a fim de determinar que informação de contexto pode ser derivada deste. Para isso, o fornecedor do sensor deve também definir o atributo descrição (`description`) do sensor. Estes atributos integram a interface de definição do sensor do ambiente de desenvolvimento ISAMadapt (Capítulo 4).

Um mecanismo de descoberta de sensores (implementado pelo `discoverer`) permite à aplicação determinar que contexto já existe no ambiente, e adicionar novos, se for o caso. Este mecanismo é útil na fase de projeto da aplicação. Para uma maior produtividade, o projetista deve gastar mais tempo determinando as situações onde a aplicação poderá executar (elementos de contexto e possibilidades de estado), e mais precisamente, qual informação de contexto é relevante e como reagir a ela, do que projetando sensores.

B.3.2 Informação Contextualizada e Personalizada

Outro serviço é refinar e transformar a faixa de dados sensorados em dados de contexto - informação de mais alto-nível. Para tal, confia-se no serviço de **agregação e tradução**. A agregação, útil em alguns tipos de contexto, faz o tratamento estatístico ou aglutinador de dados sensorados, tais como média, máximo, mínimo, centro geométrico. Estes dados são passados ao tradutor, responsável por transformar a informação em mnemônicos utilizados na programação da aplicação.

Uma abordagem simplificada para este processo prevê um conjunto mínimo de operadores que convertem de um tipo para outro. Esses operadores podem ser compostos para formar conversores mais poderosos. Por exemplo, transformar dados de GPS (altitude, latitude, longitude) em CEP, e CEP em posições de localização como casa, escritório, rua. Na definição de um elemento de contexto, o projetista faz as associações de como conseguir a informação a partir dos operadores básicos.

O desafio futuro, tratado no projeto *contextS*[14], é construir uma massa crítica de operadores básicos, gerais o suficiente para comportar um grande número de domínios de aplicações. Outra questão é definir padrões para que os operadores básicos possam ser conectados uns com os outros. Esta vem da necessidade de construir caminhos automaticamente, ou de selecionar entre várias possibilidades para um elemento de contexto gerado através da combinação de outros. Esta funcionalidade, neste momento, é de responsabilidade do programador que deve implementar tal algoritmo e instalá-lo no serviço de agregação disponibilizado.

O serviço de **predição** é outra necessidade de uma faixa de aplicações móveis que objetivam uma adaptação preventiva ou antecipativa. Por exemplo, o servidor pode determinar o comportamento futuro do usuário móvel, como atividade que costuma executar em determinada localização e horário. Desta forma, o *middleware* de execução pode fazer uma busca antecipada (*prefetching*) de arquivos que são utilizados pelo usuário naquela localização e horário, ou fazer a carga antecipada do Ambiente Virtual do Usuário para a `EXEHDAbase` a qual o usuário possivelmente se conectará. Isto conduz a trabalhar com incertezas.

B.3.3 Disseminação das Informações de Contexto

Um recurso pode ser monitorado ou por demanda ou continuamente. Segundo Ranganathan, Acharya e Saltz (1997), o monitoramento por demanda é útil em três tipos de situações: se o recurso em questão é usado com baixa freqüência e seu custo é alto; se a freqüência de alteração da disponibilidade do recurso é baixa; se o recurso é caro para monitorar. Já o monitoramento contínuo é útil se o recurso é freqüentemente usado, altera-se freqüentemente, ou é barato para monitorar.

Monitoramento por demanda é síncrono. Já, o monitoramento contínuo suporta ambas interfaces: síncronas e assíncronas. A interface síncrona é apropriada, por exemplo, para uma aplicação que envia uma seqüência de grandes mensagens e verifica o estado da rede antes de enviar cada uma. Considerando a requisição assíncrona, o sistema monitora o recurso e notifica a aplicação somente quando a disponibilidade do recurso não mais satisfizer o predicado específico.

Outra escolha no projeto do monitor de recursos é se as informações são requeridas por demanda (*polling*) ou se elas são notificadas assincronamente (*callbacks* ou eventos). A escolha depende: (a) do custo relativo do *polling* e da notificação; (b) a

[14] contextS – um *middleware* para o desenvolvimento de aplicações sensíveis ao contexto. Fonte finaciadora: CNPq, FINEP e SEPIN (edital PDI&TI/2002). Período de desenvolvimento: janeiro/2003 a dezembro/2004. Equipe integra três universidades: UFSM, UCPel, UFRGS.

freqüência com a qual as aplicações necessitam da informação; (c) se a informação corresponde a recursos gerenciados pelo sistema básico. *Pooling* é implementado de forma mais simples que a notificação, porém existem situações onde a notificação pode ser preferível, tais como no caso do ambiente móvel onde não há previsibilidade das alterações nas condições ambientais.

Desta forma, pelas características das aplicações ISAMadapt, optou-se por disponibilizar o **monitoramento contínuo com notificação de forma assíncrona**. Para adquirir a informação de contexto, a aplicação poderá fazer consulta ao `ISAMcontextService`, ou receber notificação quando da alteração na janela de monitoração, se a aplicação inscrever-se no Serviço.

Salienta-se que o ambiente móvel com alta escalabilidade necessita de um modelo de entrega de dados mais abrangente por parte do Serviço. Futuramente, o `ISAMcontexService` entregará suas informações via difusão de dados, propagando dados no ar que serão capturados seletivamente pelos clientes. Uma proposta desta forma de disseminar dados para clientes móveis foi defendida por Imielinski (1997), Acharya (1998) e Datta (1999).

B.4 Efetividade da Adaptação

É importante salientar que a efetividade da adaptação depende de muitos fatores relativos ao laço de obtenção de informação. O primeiro é quão rapidamente a aplicação é capaz de aprender sobre as alterações nas condições do ambiente – agilidade do sistema. O segundo é a exatidão das informações. Informações inexatas podem comprometer o desempenho, pois a aplicação se adapta desnecessariamente. O terceiro é o custo associado com a adaptação.

A fim de se adaptar ao ambiente móvel, um sistema deve reagir às alterações significativas tão rápido quanto elas ocorram – o sistema deve ser tão ágil quanto possível (NOBLE, 1998). Para alcançar agilidade, a linguagem deve fornecer mecanismos que permitam aos programas reagir rapidamente aos eventos assíncronos, como alteração na qualidade do serviço ou conectividade à rede. Dois mecanismos são requeridos. Primeiro, a habilidade de receber o evento de forma assíncrona (sinais, eventos) e segundo, a habilidade de tomar ações apropriadas em resposta ao evento, incluindo migração da execução para sítios diferentes.

Entretanto, a agilidade pode comprometer a estabilidade do sistema – habilidade do sistema de ignorar alterações transientes. Claramente, maximizar a agilidade e, possivelmente, sacrificar a estabilidade não é aceitável em todas as situações. Segundo Noble (1998), usuários são tolerantes a alterações freqüentes com pequenas diferenças percentuais. Em seu sistema Odyssey, Noble usa o conceito de fidelidade da imagem a ser transmitida. A alteração na banda da rede faz com que altere o grau de fidelidade. Cada grau de fidelidade fornece a mesma razão de frame, porém com diferentes qualidades dos *frames*: alta qualidade de cor, baixa qualidade da cor, tons de cinza. O rápido chaveamento entre esses níveis de fidelidade pode levar a uma experiência desagradável para o usuário. O usuário pode preferir ficar com baixa fidelidade que se expor a tais trocas, muito mais quando usuários preferem sistemas com tempo de resposta previsível, do que com latência variável, ainda que o tempo de resposta médio seja maior. Portanto, tolerância às trocas, e conseqüente necessidade da estabilidade, depende dos tipos de dados e da aplicação que faz uso deles. Além disso, quando a estabilidade é requerida, esta deve ser fornecida no contexto individual da aplicação.

Considerando os resultados da pesquisa de Noble, optou-se por atribuir ao programador da aplicação o tratamento da agilidade. O sistema deve reagir às alterações

rapidamente, porém em situações onde as alterações são muito freqüentes, o sistema deve definir faixas de valores que quando extrapolados disparam o processo de adaptação. Cabe ao programador definir as **faixas de valores** sob as quais sua aplicação tem condições de se adaptar. Esta definição é feita no menu `Context` da IDE ISAMadapt. Portanto, no ISAMadapt o balanceamento entre a agilidade da adaptação e a intrusividade que esta causa está a cargo do projetista da aplicação, que deverá conhecer o ambiente de uso da aplicação e o usuário

Outro ponto importante para projetar a adaptação é se conhecer a **exatidão da informação** fornecida pelo sensor. Se a informação é altamente variável a aplicação pode adotar uma adaptação mais conservativa, enquanto que uma adaptação mais otimista pode ser adotada se a informação é exata.

Como as aplicações podem estar interessadas em diferentes tipos de informações, o servidor deve fornecer um rico conjunto de informações (DEWITT et al., 1997). A diversidade resulta dos diferentes tipos de informações que são gerados por entidades diferentes, mantidas em diferentes formatos e localizações, e se alteram em diferentes escalas de tempo. Por outro lado, diferenças significativas advêm da forma de coleta dessas informações e de como elas são disponibilizadas.

Aplicações podem ser sensíveis à variação em mais de um elemento de contexto. E, neste caso, pode ocorrer de a alteração em ambos elementos de contexto acontecer ao mesmo tempo. Por questão de simplificação, o modelo atualmente não equaciona o problema de conflitos entre variações de contexto, mas tenta impedir sua ocorrência. É assumido que as variações não ocorrerão ao mesmo tempo. Para evitar problemas de corretude na aplicação, adota-se o conceito de **importância** para o contexto. Em caso de conflito, esta é a métrica de seleção que permite ao `ISAMcontextService` decidir qual elemento de contexto tem prioridade e notificar a aplicação de sua alteração. O programador informa, usando a interface do menu `Context` da IDE ISAMadapt, qual a ordem de importância dos diferentes elementos de contexto para a aplicação.

O **custo da adaptação** e o ganho em desempenho são tipicamente estimados baseados em informações sobre o estado corrente e o estado ótimo. O benefício de desempenho é normalmente proporcional ao tempo, de forma que

```
Custo_adaptação < benefício = ganho X duração.
```

Infelizmente, como a duração do benefício é desconhecida, pois depende das condições futuras da rede, Steenkiste (1999) concluiu que num ambiente “de melhor esforço” não existe a garantia que a adaptação paga o seu custo em aplicações *network-aware*. Na Computação *Pervasiva*, a adaptação é inerente ao tratamento da heterogeneidade e dinamicidade do ambiente, e não é somente uma estratégia para melhoria de desempenho. Portanto, o custo da adaptação é uma questão que merece uma avaliação diferenciada no ambiente consciente do contexto da Computação *Pervasiva*.

B.5 Organização Estrutural do *ISAMcontextService*

A arquitetura divide a responsabilidade entre sensor, `ISAMcontextService` e aplicação. O papel de cada um é bem definido. Para manter a independência entre estes componentes, a comunicação entre eles baseia-se em contatos expressos através de comandos escritos em XML, e na comunicação através dos multi-espaços de objetos.

Desta forma, faz-se uma clara separação entre definição de contexto – realizada pela aplicação; medida de contexto – realizada pelos sensores; interpretação, gerenciamento e disseminação do contexto – realizados pelo `ISAMcontextService`. Esta separação é refletida na organização da arquitetura do serviço.

A Figura B.1 mostra a trajetória dos dados sensorados dentro da arquitetura. Os dados ao entrarem, através do monitor, recebem um tratamento cujo fluxo é baseado no modelo *pipeline*. Os principais componentes da arquitetura são: monitor, coletor, defletor, agregador, tradutor, preditor e notificador. A funcionalidade destes componentes é descrita a seguir.

B.5.1 Coletando Informações: a Relação sensor-monitor-coletor

Para o monitoramento do contexto ISAM, não se está interessado em adaptação somente em resposta a alterações na qualidade da comunicação, mas como conseqüência de alterações em uma larga faixa de fatores ambientais (localização física, disponibilidade de energia, capacidade dos dispositivos, etc...), fatores funcionais (comunicação entre componentes, acesso a recursos, etc...) e fatores pessoais (perfil e preferências do usuário). Coletar essas informações ainda é um desafio a ser aprofundado no projeto contextS.

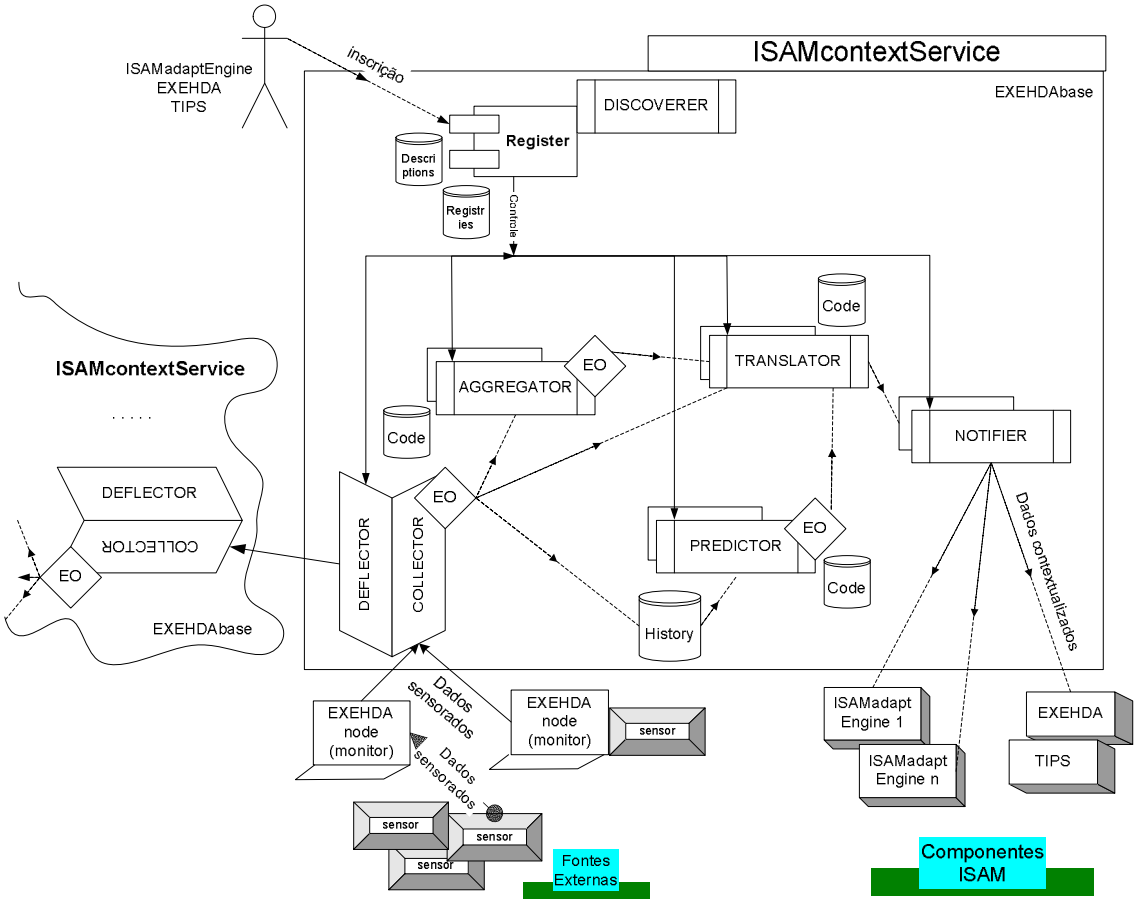


Figura B.1: Componentes do `ISAMcontextService`

As informações de contexto necessárias para as aplicações podem ser agrupadas em categorias: estáticas, dinâmicas e previsões futuras. A estimativa de comportamento futuro é certamente a tarefa mais complexa e requer um suporte sofisticado. Se a rede suporta reserva de recursos, o status da reserva pode ser usado como guia para o uso futuro. O conhecimento do comportamento do usuário e de padrões de uso da execução da aplicação, e estimativas de quanto tempo elas executam pode também guiar o uso futuro. Entretanto, este é ainda um problema desafiador (BANAVAR; BERNSTEIN, 2002). Prevê-se a utilização de Redes Bayseanas para o tratamento da incerteza nos

dados da monitoração para estimar o comportamento futuro de utilização de recursos. Este tema está começando a ser abordado no subprojeto TIPS (REAL, 2003).

A aquisição de informações estáticas e dinâmicas do ambiente é função dos componentes sensor, monitor e coletor do `ISAMcontextService`.

B.5.1.1 Detalhes sobre o Sensor

Para separar o componente que adquire dados do ambiente, e permitir o uso de diversas tecnologias de sensoramento, os **sensores são externos ao Serviço**. Note que todas as questões relacionadas ao monitoramento devem ser resolvidas no nível do sensor, pelo seu fornecedor. O `ISAMcontextService,` propriamente dito, não interfere nessas decisões. Isto elimina grande parte da complexidade associada ao Serviço, e permite a flexibilidade e generalidade exigida pelo ISAM. Desta forma, cada sensor pode ter requisitos próprios e dependentes do dispositivo/elemento que monitoram.

Como cada sensor pode ter especificidades próprias, as informações necessárias para sua ativação/desativação são disponibilizadas no sistema através do processo de registro (*registrySensor*), executado pelo fornecedor do sensor como requisito para tornar o sensor disponível no sistema. Essas informações, expressas no formato XML, consistem em definir: (a) que dados o sensor pode fornecer – *description*; (b) qual o formato dos dados a serem enviados ao coletor; (c) onde o sensor está localizado - *location*; (d) como comunicar-se com ele - *activation*. A comunicação é estabelecida através da rotina de ativação do sensor, escrita em Java, que será executada pelo Serviço quando requerido que o sensor forneça dados ao sistema. Esta rotina usa como parâmetro de entrada a informação, fornecida pelo `ISAMcontextService`, do número da porta para comunicação com o monitor.

O Serviço usa o componente monitor como representante do sensor no sistema. Através do processo de registro, o sistema sabe a localização do monitor que gerencia e representa o sensor. Como muitas aplicações podem estar fazendo parte da mesma localidade espacial (`EXEHDAbase`), a organização do Serviço permite que, em vez de disparar todo o processo de sensoramento, a aplicação pode fazer uso de informações já disponíveis. Para isto é requerido conhecer e representar a localização do sensor (nodo hospedeiro). Esta localização é armazenada, junto com as demais informações de configuração do sensor, usando o Serviço de Diretórios do `register`, implementado com o protocolo LDAP (*Lightweight Directory Access Protocol*).

Alguns sensores fornecem informações também para o gerente de execução EXEHDA, e serão acionados por este quando do início de suas atividades, e terminarão suas atividades quando este terminar.

B.5.1.2 Detalhes do Monitor

Existe um monitor em cada máquina que tem um sensor conectado. Um monitor pode estar associado com vários sensores, e sua função é servir de elo entre o servidor e os sensores registrados. O monitor recebe as informações no formato XML e as envia ao coletor, localizado na máquina-base, que as disponibiliza no espaço de objetos para os demais componentes. A organização hierárquica entre monitor e coletor visa minimizar a troca de mensagens e a sobrecarga do componente na máqina-base.

Outra função do monitor é gerenciar os sensores: ativar e desativar os sensores, suspender e retornar a execução dos sensores.

Como visto anteriormente, pelas características das aplicações ISAM o **monitoramento será contínuo com notificação assíncrona**, a partir da ativação do sensor e da inscrição da aplicação no Serviço.

Para o monitoramento contínuo, uma decisão importante é a **granulosidade da monitoração**. A alternativa mais simples é registrar cada alteração. Esta escolha pode ser impraticável em muitas situações tendo um alto impacto no desempenho da aplicação (LOWEKAMP et al., 1998). Outra alternativa é usar um valor limite e rastrear somente as alterações que extrapolam esse limite. Esta escolha é mais adequada a recursos estáveis, com poucas alterações em seu estado. Nestes casos, alterações transientes podem ser tratadas com filtros que as eliminam. O serviço deve permitir às aplicações registrar filtros específicos (da aplicação ou do recurso) para determinar quais alterações na disponibilidade/qualidade do recurso devem ser registradas. Assim, o monitoramento pode ser parametrizado para haver coleta periódica das informações, em uma unidade de tempo.

No `ISAMcontextService` o envio de dados do sensor ao monitor é sob demanda – o monitor requisita a informação. Como várias aplicações podem requisitar dados do mesmo sensor, a freqüência de envio de dados se dará pela menor unidade de tempo requerida pelas aplicações, e é responsabilidade do monitor fazer os ajustes necessários. Por exemplo, se a monitoração de temperatura está sendo realizada a cada 1 hora, e outra aplicação solicitar monitoração a cada 10 minutos, o monitor será reconfigurado para a nova freqüência de captura de dados.

B.5.1.3 Detalhes do Coletor

Quando o monitor recebe informações do sensor, este as envia ao coletor. O processo de envio de informações do monitor para o coletor pode ser: *pooling* – o coletor requisita a informação, ou periódico – envia quando houver variação nos dados monitorados, considerando as faixas de valores estipuladas.

O sistema pode estabelecer um *timeout* relativo ao tempo que a tupla ficará no Espaço de Objetos até ser enviada ao armazenamento (arquivo *history*[15]) se a opção `storage` for requerida por uma das aplicações interessadas no elemento de contexto. Se *timeout* é zero, a tupla é enviada para a história quando chegar outra tupla, este procedimento simula a reescrita de dados de contexto, e serve para informações que somente são relevantes se atualizadas. `Storage=off` é o comportamento *default* do coletor.

Alguns dos trabalhos realizados pelo grupo sobre monitoração serão integrados ao `ISAMcontextService`. Estes abordam duas dimensões de informações: um, monitora a aplicação e outro, monitora o sistema básico. No nível de aplicação, o monitoramento é realizado usando a JVMPI (*Java Virtual Machine Profile Interface*), a qual oferece a possibilidade de seleção dinâmica de eventos de interesse da aplicação que podem ser monitorados (ativação, interrupção, tempo de espera dos métodos, etc) (ARAUJO, 2002). Já, no nível de sistema, atualmente existem dois tipos de métricas: (a) uma para sensores de recursos do nodo, e (b) outra para recursos da aplicação (SILVA, 2003).

B.5.2 Transformando Dados Sensorados em Dados de Contexto

Muitos tipos de contexto são inerentemente ambíguos (DEY, 2000). A ambigüidade está na interpretação da informação, não na sua captura. Desta forma, distingue-se entre

[15] Dados usados para o serviço de predição.

dados do sensor e dados de contexto. Os diversos componentes do Serviço manipulam e transformam os dados, numa arquitetura *pipeline*, que recebe dados em um formato e enviam em outro, até obter os mnemônicos definidos pela aplicação. Por exemplo, uma aplicação pode requerer que o valor numérico do elemento de contexto largura de banda seja traduzido como dado de contexto na forma "alta", "média", "baixa"; outra aplicação pode requerer a mesma informação na forma "suficiente", "insuficiente". A transformação de dados sensorados em dados de contexto para cada aplicação é realizada pelos componentes de conversão de informações: **agregador, preditor** e **tradutor**.

B.5.2.1 Detalhes do Agregador

Informações observadas podem não ser significativas se consideradas de forma isolada. Assim, o agregador (`aggregator`) permite manipular os dados através de funções estatísticas como média, contador, máximo ou mínimo, ou funções de conversão de formatos, como graus Celsius em Fahrenheit. A agregação também permite compor informações de mesmos elementos de contexto, como a seleção de objetos que estão na mesma localização usando uma relação de vizinhança, ou de diferentes elementos de contexto (relativos a uma entidade), como deduzir a atividade do grupo (reunião, encontro, apresentação) baseado nas informações de pessoas presentes e aplicação que cada uma executa, transformando-as em outra informação.

A aplicação pode selecionar agregadores disponibilizados pelo sistema (nativos) ou criar um próprio. Novos agregadores devem ser registrados no sistema através da interface do componente `register: registryAggregator, removeAggregator`. A remoção somente é permitida se a aplicação é dona do agregador, isto é, se ela o registrou.

Existe um agregador principal na instância do Serviço. Este fica num laço recebendo informações de controle do `register`. Quando recebe a mensagem de ativação, cria uma nova *thread* para executar o algoritmo de agregação da aplicação. Assim, existe um agregador para cada elemento de contexto de cada aplicação. Quando recebe a mensagem para desativação, a *thread* correspondente é encerrada.

As informações geradas ficam disponíveis um certo tempo (*timeout* dependente do tipo de sensor) no espaço de Objetos do agregador, sendo retiradas para outro espaço (arquivo *History*) o qual pode ser utilizado pelo preditor, se este procedimento foi solicitado pela aplicação (`storage=on`).

B.5.2.2 Detalhes do Preditor

O preditor (`predictor`) pode utilizar diferentes algoritmos de previsão de tendência ou comportamento dos dados, como Redes Bayseanas (JENSEN, 1996), sobre os dados históricos armazenados no sistema. Outra função dos dados históricos é disponibilizar dados para consulta da aplicação: `queryHistorybyDate(dates), queryHistorybyTime (times), queryHistorybyPeriod (data, times)`.

Existe um preditor principal na instância do serviço. Este fica num laço recebendo informações de controle do `register`. Quando recebe a mensagem de ativação, cria uma nova *thread* para executar o algoritmo de previsão da aplicação. Assim, existe um preditor para cada elemento de contexto de cada aplicação. Quando recebe a mensagem para desativação, a *thread* correspondente é encerrada.

B.5.2.3 Detalhes do Tradutor

Como as aplicações utilizam dados de contexto codificados em mnemônicos (alto nível), a tradução dos dados para os mnemônicos é realizada pelo tradutor (`translator`), e disponibilizadas no espaço de objetos do notificador (`notifier`).

O tradutor acessa o espaço de objetos do coletor, do agregador, do preditor ou do próprio tradutor como entrada. As regras de tradução disponibilizadas inicialmente são simples e prevêem operações da forma: operador – operando - gatilho. Operandos são os valores do coletor, preditor ou agregador. Operadores são relativos às operações: maior (`valueUp:` valor obtido >= `operando1`), menor (`valueDown`: valor obtido <= `operando1`), limites (`valuei – valuef: operando1` <= valor obtido <= `operando2`), vizinhança (`around:` valor obtido difere para +/- na unidade do `operando1`) e tabelas (`table-search`: valor obtido é usado como índice na tabela de nome `operando1`), os dados da tabela foram informados com o operador `table-entry` onde `operando1` é o nome da tabela e `operando2` é o valor. Se estas regras não forem suficientes para expressar a tradução, o projetista pode fornecer o código para a tradução (operador: `other`).

Por exemplo, se a CPU disponível está acima de 10Mb o valor é traduzido para "load", se estiver abaixo de 2Mb é traduzido para "unload". No formato XML, as regras de tradução para o elemento de contexto CPU ficam:

```
<rule>
   <valueUp> 10Mb </valueUp>
   <trigger> load </trigger>
</rule>
<rule>
   <valueDown> 2Mb </valueDown>
   <trigger> unload </trigger>
</rule>
```

Uma solução mais elaborada é necessária, a qual gerará uma linguagem baseada em XML para descrição das regras de tradução e respectivo interpretador, e integra um dos pontos de pesquisa do projeto contextS.

Operacionalmente, existe um tradutor principal na instância do serviço. Este fica num laço recebendo informações de controle do `register`. Quando recebe a mensagem de ativação, cria uma nova *thread* para executar a tradução do contexto para a aplicação. Assim, existe um tradutor para cada elemento de contexto de cada aplicação. Quando recebe a mensagem para desativação, a *thread* correspondente é finalizada.

B.5.3 Notificando as Alterações

O notificador (`notifier`) usa a estrutura de dados de uma lista de aplicações interessadas em receber a notificação de alteração de estado em um elemento de contexto. Portanto, existe uma lista para cada elemento de contexto subscrito no Serviço. As informações armazenadas por aplicação são: referência a `ISAMadaptEngine` da aplicação, lista de possibilidades de estado do contexto (mnemônicos), último estado notificado (`lastNotified`).

O mecanismo de notificação do Serviço é baseado em tupla de natureza reativa. Esta solução mantém o assincronismo da comunicação e permite o tratamento da desconexão pela aplicação. O componente notificador, quando da entrada de uma tupla no seu espaço de objetos, escrita pelo tradutor (`translator`), verifica a necessidade de notificar a aplicação da alteração de contexto: se o elemento é um dos possíveis estados definidos pela aplicação, e se é diferente do notificado anteriormente (houve troca de

contexto). A lista de aplicações para o elemento de contexto é percorrida e todo evento relevante é notificado.

Operacionalmente, no sistema existe um notificador principal e uma *thread* para cada aplicação. Cada *thread* é sincronizada com o correspondente tradutor para realizar o processo de notificação.

B.5.4 Obtendo as Informações entre Células

O `deflector` é o responsável por disseminar as informações coletadas localmente para outras células ou componentes interessados nessas informações, como o escalonador TIPS (REAL, 2003). Essas informações também poderão compor um elemento de contexto de dimensão global. O `deflector` obedece aos comandos de controle do `register` que lhe configura para enviar (*multicast*) às demais instâncias do Serviço as informações coletas sobre um dado elemento de contexto. A instância local do Serviço, onde o contexto global foi requerido, é a responsável pelo gerenciamento dessas informações. Um algoritmo de agregação irá determinar o contexto global a partir dos dados recebidos pelo coletor, e estes serão processados da forma descrita anteriormente.

B.5.5 Interagindo com a Aplicação

O `register` é o componente que implementa a interface com o usuário, aplicação/`ISAMadaptEngine` ou os serviços do *middleware* EXEHDA, através dos serviços de inscrição e consulta. O serviço de descoberta de recursos é implementado pelo `discoverer`.

B.5.5.1 Serviço de Inscrição

A aplicação registra seu interesse em receber notificações na alteração do estado de um elemento de contexto através da rotina `subscribe(elementContext, ISAMadaptEngine_ref, descriptorContextFile)`enviada ao componente `register`. Este reconfigura o Serviço para os parâmetros fornecidos pela aplicação, enviando mensagens de subscrição aos componentes do Serviço. Se necessário, é reconfigurado o monitor/coletor relativo ao parâmetro freqüência de envio de dados; o agregador é criado e ativado (um para cada elemento de contexto da aplicação); o preditor é criado e ativado, se for o caso; o tradutor é criado e ativado, o notificador é criado e ativado.

Para simplificar a implementação, é imposta a restrição de que todos os entes da aplicação usam a MESMA definição do elemento de contexto. Na implementação atual não há o controle de redefinição para a noção de contexto individualizado por ente. Observe que um novo processo (*thread*) agregador, preditor, tradutor, notificador é criado a cada novo elemento de contexto de interesse da aplicação. Isto se deve ao fato de que cada aplicação pode ter uma forma diferente de tratar e usar os mesmos dados sensorados para o mesmo elemento de contexto. `Reflector`, `register`, `discoverer` e `collector` são processos únicos no Serviço (instância local).

Quando uma aplicação termina, `unSubscribe(elementContext)` é executado pelo `register`, e a mensagem de remoção da inscrição é enviada aos componentes, os quais removem as entradas em suas estruturas de controle relativas àquela aplicação.

A aplicação pode suspender/retomar o recebimento de notificações através do uso das rotinas: `suspendNotify(contextElement)` e `resumeNotify(contextElement)`.

Porém, este procedimento não é propagado e não altera o recebimento de dados do sensor-monitor-coletor. Isto se deve ao fato de que o contexto deve ser continuamente monitorado para estar atualizado. Embora a aplicação tenha suspendido a adaptação automática, ela pode ainda querer consultar o contexto sob demanda.

O componente `register` é também responsável por atender as consultas sob demanda. Por exemplo, o `register` implementa a rotina `searchSensor`, que permite consultar os sensores nativos e/ou ativos.

B.5.5.2 Serviço de Descoberta de Recursos

A descoberta de recursos é um serviço importante para a computação *pervasiva*, pois não é garantido que todos os ambientes ofereçam os mesmos recursos/serviços para o usuário móvel. No cenário *pervasivo* não deve ser necessário instalar antecipadamente *drivers* ou software de acesso aos recursos/serviços. As aplicações podem usar a descoberta de recursos para outros propósitos, como encontrar os objetos presentes num local: consulta de sensores dos elementos de contexto (que representam os objetos) e escopo: *around*, por exemplo. Algumas aplicações podem requerer informações sobre sensores, ou outros componentes em execução, ou recursos disponibilizados. Por exemplo, se uma aplicação deseja se mover para um determinado contexto, esta usa o serviço de descoberta para indicar seu itinerário (destino da migração).

O processo de descoberta é realizado através do componente `discoverer` que procura o recurso/serviço requerido e informa à aplicação. O processo de descoberta é ativado por demanda (a) da aplicação, (b) de outro componente do Serviço, ou (c) pelo *middleware* de execução EXEHDA.

Um mecanismo de descoberta de recursos, em geral, implementa um protocolo de registro de recursos/serviços e um protocolo de descoberta destes (GUPTA; TALWAR; AGRAWAL, 2002). O modelo para a descoberta de serviços divide-se em dois níveis. O nível interface consiste num protocolo de descrição de recursos/serviços, usando XML, através do qual as aplicações (`ISAMadaptEngine, EXEHDA/TIPS`,...) interrogam sobre a localização dos recursos/serviços. O nível implementação consiste num processo responsável por: (a) receber as requisições, enviá-las para as instâncias do *discoverer* onde serão processadas, e retornar o resultado à aplicação; (b) registrar e procurar os recursos/serviços disponibilizados na arquitetura. Esta divisão garante que os desenvolvedores de aplicações têm uma interface comum para consulta e que os fornecedores de serviços sabem como disponibilizar os serviços. Um estudo para projetar este serviço no `ISAMcontextService` está sendo realizado como tema de dissertação de mestrado (SCHAFFER, 2004), sob a ótica da computação em grade, e de monografia de especialização (FONTOURA, 2003) sob a ótica da computação móvel.

B.5.6 Movimento dos Componentes da Aplicação

Como os entes da aplicação podem mover-se para outros locais físicos (`EXEHDAnode`), o contexto em que eles estão inseridos pode alterar-se. Quando um ente deixa o `EXEHDAnode`, a rotina de `unSubscribe` é executada para retirar todas as entradas da aplicação no Servidor local. No novo serviço local, o qual hospeda o ente que se movimentou, são inseridas as informações relativas ao contexto através da execução da rotina `subscribe`. A partir daí, a aplicação pode se adaptar ao novo contexto determinado pela sua nova localização física. Esta semântica está embutida na implementação do comando `move` da linguagem (Capítulo 4).

Para simplificar este processo de migração, a descrição do contexto de interesse dos entes da aplicação é armazenada num arquivo da base de armazenamento *pervasivo* (`ISAMbda`), permitindo o gerenciamento de sua localização pelo sistema.

B.5.7 Organização Distribuída

O `ISAMcontextService` está sob a gerência do *middleware* de execução EXEHDA. Em geral, cada `EXEHDAbase` contém os componentes da arquitetura desse serviço, e monitora o contexto na sua área de cobertura. O Serviço está organizado de forma distribuída, refletindo a estrutura organizacional do *middleware* de execução EXEHDA.

Nos nodos móveis ou fixos da rede executam, em geral, os monitores, que estão associados aos sensores que executam nos nodos-folhas da arquitetura. Esta parte reflete uma arquitetura altamente distribuída. Monitores se reportam aos coletores, que estão na parte fixa da rede, em nodos intermediários ou na EXEHDAbase. Por questões de escalabilidade, componentes relativos aos dados sensorados têm uma organização altamente distribuída, enquanto que componentes relativos aos dados de contexto estão centralizados na EXEHDAbase, uma vez que potencialmente atendem a várias aplicações. A Figura B.2 ilustra uma organização espacial possível da distribuição do ISAMcontextService.

Porém, a probabilidade de desconexões dos nodos móveis e questões de desempenho induzem a necessidade de tratamento local dos dados relativos aos elementos de contexto local. Desta forma, uma instância leve do Serviço executa no nodo móvel, quando os entes da aplicação necessitarem de informações sobre o contexto local.

Assim, a organização distribuída dos componentes do Serviço de Reconhecimento de Contexto é dinâmica, e decidida pela dimensão dos elementos de contexto que a aplicação necessita: local, ou celular.

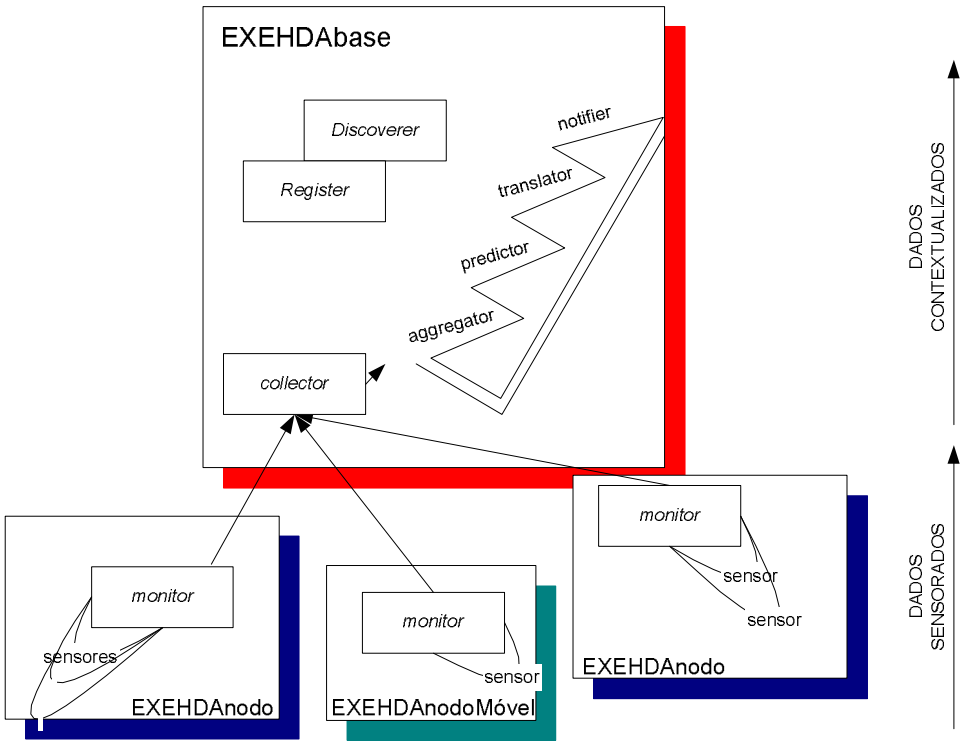


Figura B.2: Exemplo de Distribuição dos Componentes `ISAMcontextService`