ISAMadapt - um Ambiente de Desenvolvimento de Aplicações para a Computação Pervasiva

Iara Augustin[1]*, Adenauer Yamin[2], Luciano Cavalheiro da Silva[3], Rodrigo Real[3], Cláudio Resin Geyer[3]

[1]Centro de Tecnologia, Universidade Federal de Santa Maria (UFSM)
Campus Universitário, Faixa de Camobi km 9,
Santa Maria, RS, Brasil

[2]Escola de Informática, Universidade Católica de Pelotas (UCPel),
R. Félix da Cunha 412, Pelotas, RS, Brasil

[3]Instituto de Informática, Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 - 91.501-970 - Porto Alegre - RS - Brazil

august@inf.ufsm.br,adenauer@atlas.ucpel.tche.br
{lucc,rreal,geyer}@inf.ufrgs.br

Abstract. *ISAMadapt is a development environment to mobile applications with adaptive behavior in a pervasive computing environment. To make available the follow-me semantics the pervasive applications are distributed, mobile and adaptive to the execution context. The main abstractions to build applications for this scenario are: context, alternative behaviors, commands and adaptation policies. The application becomes context-aware, and adapts to it, through a process that involves two phases: conception and execution. Contextualized and adaptive commands were added to a language-base and the middleware, called EXEHDA, implements the dynamic functionality of the abstractions and manages the pervasive environment.*

Resumo. *ISAMadapt é um ambiente de desenvolvimento de aplicações móveis com comportamento adaptativo ao contexto em um ambiente de Computação Pervasiva. Para disponibilizar a semântica siga-me, as aplicações pervasivas são distribuídas, móveis e adaptativas ao contexto em que executam. As principais abstrações para produzir aplicações para este cenário incluem: contexto, comportamentos alternativos, comandos e políticas de adaptação. A aplicação torna-se consciente de seu contexto, e adapta-se a ele, através de um processo que envolve duas fases: concepção e execução. Comandos relativos às abstrações foram adicionados a uma linguagem-base, e o middleware, chamado EXEHDA, implementa a funcionalidade dinâmica dessas abstrações e gerencia o ambiente pervasivo.*

1. Introdução

Um novo cenário computacional, chamado *Computação Pervasiva* (*Pervasive Computing*), está sendo considerado o novo paradigma do século 21 [Saha and Mukherjee, 2003, Satyanarayanan, 2001]. Este cenário prevê uma mobilidade física (equipamentos e usuários) e uma mobilidade lógica (componentes da aplicação e serviços) em escala global. Ao usuário final é necessário fornecer transparência dessa mobilidade de forma

*Pesquisa parcialmente financiada pelo CNPq/SEPIN/FINEP.

que este possa acessar seu ambiente computacional independente de sua localização e do equipamento que está utilizando. Para disponibilizar esse tipo de aplicação é necessário uma infra-estrutura de suporte para sua construção e execução. Este é o objetivo do projeto ISAM (Infra-estrutura de Suporte às Aplicações Móveis Distribuídas, www.inf.ufrgs.br/~isam) [Augustin, 2004, Augustin et al., 2003, Augustin et al., 2002a, Augustin et al., 2002b, Yamin et al., 2004, Yamin, 2004]. A tese defendida é que a infra-estrutura de suporte à *pervasividade* em escala global pode ser construída através da integração de três áreas da computação: Computação Móvel, Computação em Grade e Computação Consciente do Contexto [Yamin et al., 2003]. Esta visão é diferente da de outros projetos de infra-estrutura para a *Computação Pervasiva*, tais como Aura [Garlan et al., 2002] e Gaia [Roman, 2003], que propõem soluções focadas em uma visão de computação pessoal e em uma visão de contexto local, respectivamente.

Nossa proposta de sistema de suporte para esse ambiente usa a metáfora de um ambiente virtual do usuário, onde as aplicações têm o estilo siga-me (*follow-me applications*). A arquitetura ISAM fornece um ambiente de desenvolvimento e de execução como suporte às aplicações conscientes da mobilidade, o qual permite ao programador tirar vantagem das características dinâmicas do ambiente *pervasivo* sem consumir muito de sua atenção.

A arquitetura ISAM é abrangente [Augustin, 2004, Yamin, 2004]. Por questões de escopo, este artigo somente introduz as abstrações para expressar um comportamento adaptativo ao contexto sob a ótica da programação. O artigo está estruturado da seguinte forma. A seção 2 introduz o escopo de pesquisa. A seção 3 identifica as abstrações a serem inseridas numa linguagem de programação a fim de permitir expressar um comportamento adequado a este novo ambiente computacional. A seção 4 apresenta uma aplicação-exemplo para ilustrar como as abstrações foram implementadas na linguagem ISAMadapt. Na seqüência, a seção 5 apresenta rapidamente o processo de tradução do código ISAMadapt para Java. Os trabalhos relacionados são analisados na seção 6. As conclusões são apresentadas na seção 7.

2. ISAM: Software-Âncora para a Computação *Pervasiva*

O foco da pesquisa atual do projeto ISAM é o estudo das questões envolvidas na modelagem e execução de aplicações móveis, distribuídas e conscientes do contexto em um ambiente de *Computação Pervasiva*. A mobilidade do usuário requer um novo modelo de aplicações que vê os elementos da computação espalhados 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 software permanentemente para executá-lo, como ocorre hoje com a tecnologia usada nas aplicações da computação móvel.

Na arquitetura ISAM, o ambiente computacional é a rede em uma grade global, referenciado como ISAMpe (*ISAM Pervasive Environment*), como ilustrado na Figura 1. Os componentes do ambiente computacional (dados, código, vários tipos de dispositivos, serviços e recursos) estão espalhados, e são gerenciados por um sistema de suporte (middleware) que fornece um acesso *pervasivo* a eles. Cada usuário tem um ambiente virtual que pode ser acessado na localização em que se encontra e com o equipamento de que 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

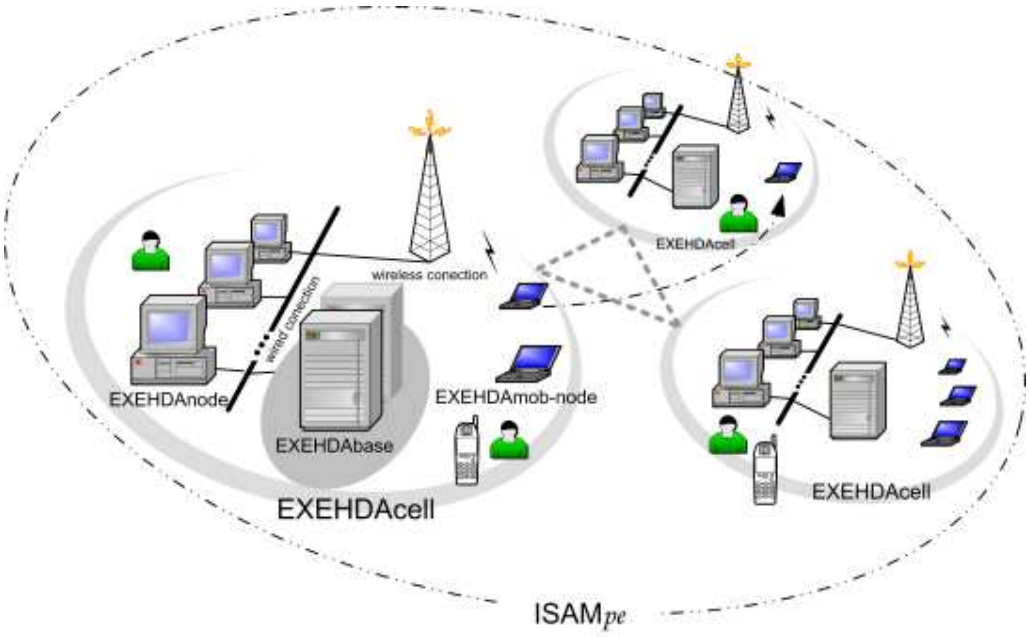


Figura 1: ISAMpe - *ISAM Pervasive Environment*

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.

Para suportar a visão *pervasiva* do ISAM foi definida uma arquitetura de software que está sendo construída [Augustin, 2004, Yamin, 2004]. Os principais componentes dessa arquitetura são: o ambiente de desenvolvimento de aplicações, chamado **ISAMadapt**; o middleware de suporte à execução e gerenciamento das aplicações, chamado **EXEHDA**; e a linguagem Java em suas versões J2SE (*Standard Edition*) e J2ME (*Micro Edition*).

O middleware disponibiliza um acesso *pervasivo* a dados e código através dos componentes ISAMpe, Ambiente Virtual do Usuário (AVU), Ambiente Virtual da Aplicação (AVA) e 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 componente 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 do ambiente *pervasivo*. O middleware também disponibiliza o Serviço de Reconhecimento de Contexto, o qual é responsável por informar o estado dos elementos de contexto de interesse da aplicação e do próprio ambiente de execução. O `ISAMcontextService` monitora o ambiente e notifica a aplicação sobre a troca de estado no elemento ao qual a aplicação se inscreveu, informando-lhe o estado corrente conforme mnemônicos configurados pela aplicação [Augustin, 2004]. Outro componente importante do middleware é a `ISAMadaptEngine`, cuja função é o gerenciamento da adaptação na arquitetura ISAM.

Nas próximas seções detalham-se as principais abstrações da linguagem ISAMadapt e seu relacionamento com o middleware EXEHDA.

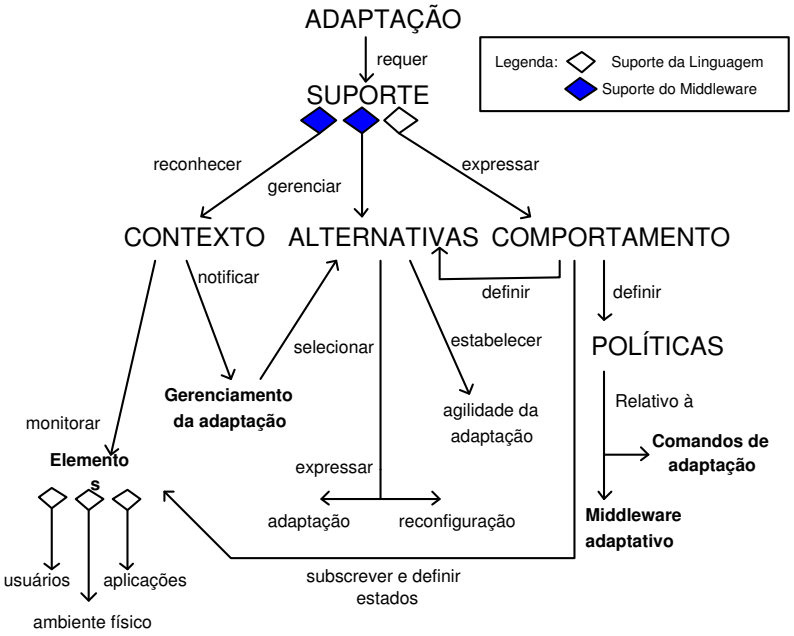


Figura 2: Modelo de adaptação

3. Abstrações Básicas da Linguagem ISAMadapt

Para o projeto e implementação de uma aplicação pervasiva é disponibilizado um ambiente que orienta o programador nas definições de configurações e na codificação da aplicação em si [Augustin, 2004]. A execução é controlada pelo middleware EXEHDA, o qual gerencia os aspectos derivados da mobilidade. Esta **integração** permite diminuir a complexidade do código da aplicação para o tratamento de adaptações dinâmicas ao contexto em que a aplicação se encontra no momento.

No ambiente pervasivo, a mobilidade física introduz a possibilidade do movimento do usuário durante a execução da aplicação. Enquanto o usuário se movimenta, os recursos acessíveis/disponíveis podem se alterar, não só em função da área de cobertura e heterogeneidade das redes, como em função de a 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. Codificar uma aplicação pervasiva é especificar qual código deve executar no ambiente em que a aplicação está no momento. Deste objetivo deriva a necessidade de **adaptação dinâmica ao contexto**.

Para implementar essa adaptação, projetou-se um modelo de adaptação [Augustin, 2004], resumido na Figura 2, que permitiu identificar as principais abstrações para escrever código adaptativo no ambiente computacional *pervasivo*. As abstrações identificadas são: **contexto, adaptadores, comandos** e **políticas de adaptação**. Cada uma dessas abstrações é implementada em dois tempos: (i) programação, baseado no suporte fornecido pelo ambiente de desenvolvimento; (ii) execução, baseado no suporte fornecido pelo middleware. A Figura 3 ilustra os principais componentes que implementam as abstrações nesses dois tempos. Uma aplicação ISAMadapt é estruturada em quatro tipos de componentes:

- entes funcionais e adaptativos, que implementam a lógica da aplicação e fazem uso de comandos e métodos adaptativos;
- adaptadores, que implementam o comportamento dos métodos e entes adaptativos correspondentes ao estado do elemento de contexto corrente;
- elementos de contexto de interesse da aplicação;

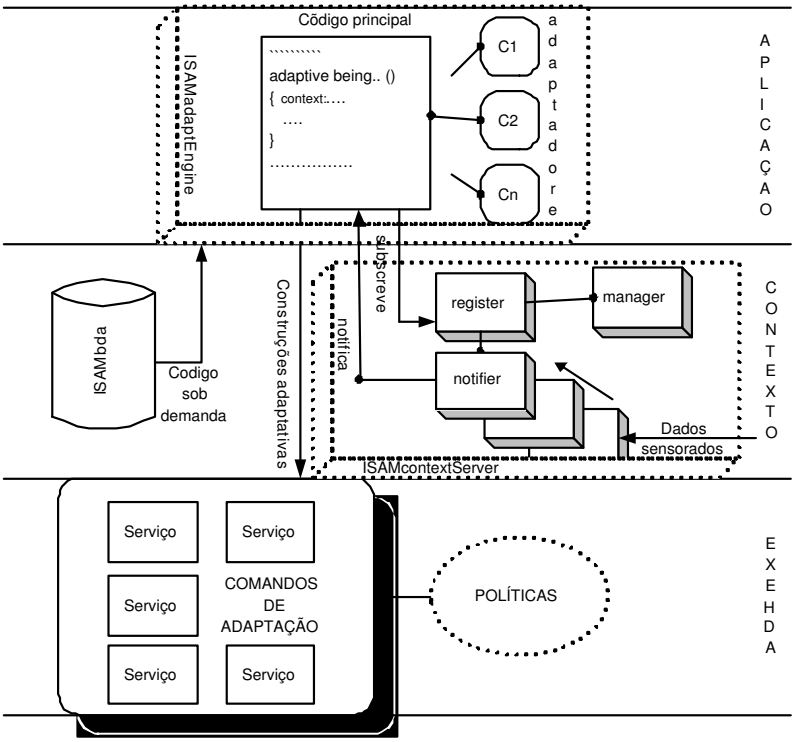


Figura 3: Componentes ISAMadapt

- políticas de adaptação, que orientam o sistema de execução nas decisões sobre a adaptação dinâmica da aplicação.

A seguir é apresentado como essas abstrações foram concretizadas em comandos da linguagem e arquivos de configuração do sistema. Essa apresentação é realizada a partir de uma aplicação-protótipo chamada Reuni@o.

4. Programando uma Aplicação

Para exemplificar a construção de uma aplicação foi escolhido o software Reuni@o (www.ufsm.br/reuniao), uma aplicação distribuída que implementa um suporte a reuniões virtuais. A reunião é representada como uma árvore de discussão, onde cada nodo é uma contribuição de um participante, como ilustrado na Figura 4. As contribuições são classificadas em tipos:

- questão: problema, dúvida ou proposição que inicia um debate ou sub-tópico de um debate;
- idéia: propõe uma solução para a questão;
- argumento: dá sustentação ou contesta uma idéia. Anexos, como arquivos e URLs, podem ser adicionados para reforçar um ponto de vista (argumento pró ou contra).

Na fase de desenvolvimento da aplicação, o programador dispõe de um ambiente de programação (`ISAMadapt IDE`) que o orienta na descrição da aplicação e do comportamento adaptativo desta. Nesta fase são codificados os vários aspectos da adaptação ao contexto: (i) definição dos elementos de contexto de interesse da aplicação; (ii) codificação dos comportamentos alternativos ajustados às condições ambientais, usando entes e métodos adaptativos e comandos de adaptação; e (iii) seleção das políticas de adaptação. A seguir, expõe-se a concepção da aplicação, a qual reflete uma metodologia embrionária de desenvolvimento de aplicações *pervasivas* [Augustin, 2004].

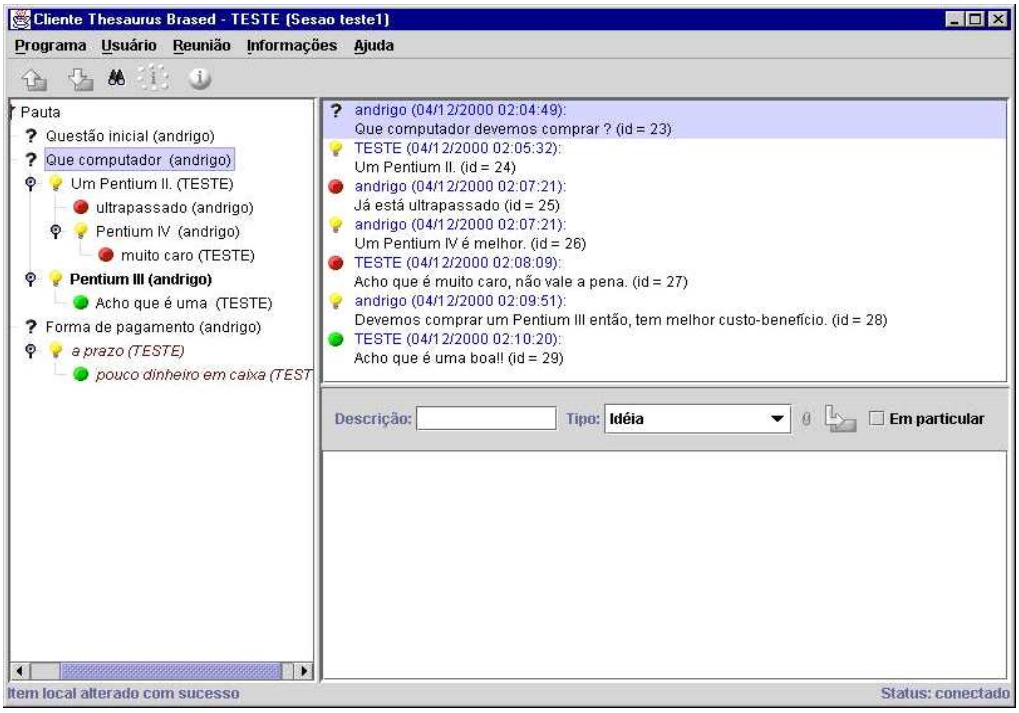


Figura 4: Uma versão da interface *desktop* do software Reuni@o

4.1. Definindo o Modelo de Contexto

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 e define seus atributos, os quais integram o contexto da aplicação, como ilustrado pela Figura 5. Por exemplo, a entidade `rede` pode ter como elemento de contexto: tipo de conexão (`wired, wireless`), estado da conexão (`connected, disconnecting, disconnected`).

Alterações no estado dos atributos dessas 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”.

Com o princípio de separação de conceitos, a especificação do contexto não está embutida no código da aplicação, mas é um componente à parte no projeto da aplicação. No ambiente de desenvolvimento `ISAMadapt IDE`, o menu `Context` permite ao programador definir e configurar os elementos de contexto de interesse da aplicação, ao qual esta poderá se adaptar. O menu 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, e gera o arquivo `context.xml`.

Considerando a aplicação Reuni@o, da qual está sendo implementado um protótipo simplificado, a visão particular de contexto é definida pelas características do dispositivo móvel em uso (`PDA, cellphone, desktop`) e pelo 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).

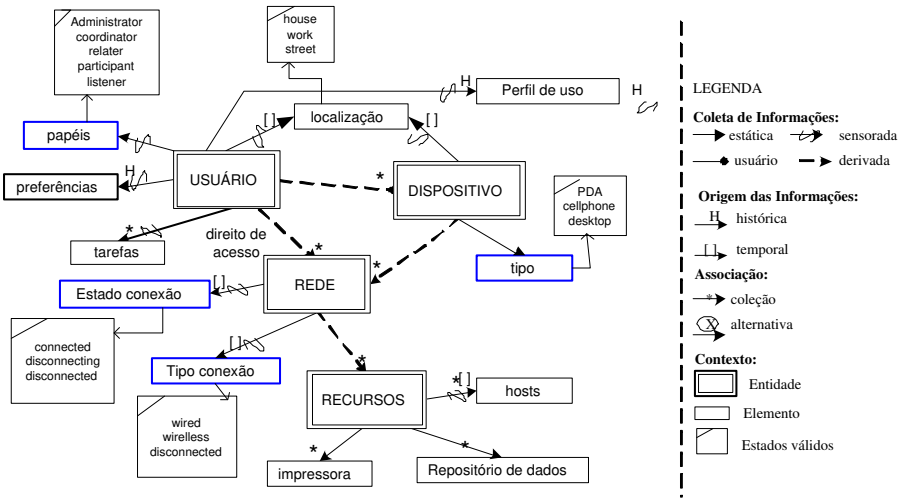


Figura 5: Exemplo de modelagem do contexto da aplicação

4.2. Definindo as Estratégias de Adaptação ao Contexto

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 podem usar: migração (comando `move`), código sob demanda (implícito à arquitetura ISAM), disseminação de dados (comando `push`), ativação/desativação de atividades devido a troca de papéis na reunião, prefetching de anexos (comando `prefetch`) e filtragem de dados para ajuste da capacidade dos recursos (ente de sistema chamado `filter`), e sincronização de buffers locais para tratamento da desconexão (comando `disconnect/reconnect`). 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.

A aplicação Reuni@o tem três grandes atividades: (i) entrar na reunião, (ii) gerar informações e (iii) receber informações dos demais participantes. A adaptação poderá ocorrer, nessas atividades, da seguinte forma:

- inicialização - a aplicação com o código adaptado ao tipo do dispositivo corrente é disparada pelo middleware. Ao iniciar a aplicação, é solicitado a seleção da reunião da qual o usuário participará, e a identificação deste para certificação (`id, senha`). Estes dados são enviados ao serviço de `login` da parte servidor da aplicação que certifica o usuário como integrante da reunião. A informação dos elementos de contexto, tipo de conexão e papel do participante, é utilizada para carregar sob demanda os demais componentes que comporão a interface gráfica da aplicação para o dispositivo corrente;
- emissão - é relativa ao envio das contribuições e dos anexos, que considera o estado do elemento de contexto tipo de conexão corrente: `wired`- alta fidelidade; `wireless`- baixa fidelidade, envio postergado (em partes); `disconnected` - buferização local para sincronização futura;
- recepção - é relativa ao recebimento das contribuições e dos anexos, que considera os elementos de contexto: tipo de dispositivo (`notebook, PDA, cellphone`) e conexão corrente (`wired, wireless, disconnected`). No `notebook` toda a árvore de discussão pode ser carregada, se com conexão `wired`; se a conexão é `wireless`, somente a parte não lida pelo participante é carregada, sendo as demais carregadas por demanda conforme a solicitação do participante; se desconectado, 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 `wired`, carga e disparo

automático do aplicativo correspondente para leitura; sob conexão `wireless`, carga e disparo do aplicativo para leitura seguindo uma ordem de prioridade definida pelo participante (perfil do usuário); desconectado, é emitida uma mensagem indicando a existência de anexos (nomes e URLs).

A informação `conectado/desconectado` está sempre presente na barra de status da ferramenta. A aplicação deve permanecer funcional mesmo sob desconexão da rede. Para tanto, é necessário manter o armazenamento local de informações, e prover mecanismos de sincronização na reconexão (esta funcionalidade é disponibilizada pela implementação do comando `disconnect/reconnect` pelo middleware). No modo conectado (`wired` ou `wireless`), um ente na rede fixa enviará as novas contribuições para o participante conforme estas sejam geradas, ou considerando o intervalo de tempo definido para o recebimento de novas informações.

A interface gráfica da aplicação oferecida ao usuário 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 gráfica, 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 destes, 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 tag correspondente ao envio de contribuições.

4.3. Codificando a Aplicação

A modelagem da aplicação é baseada na Hololinguagem (www.inf.ufrgs.br/~holo), onde a aplicação é modelada com dois tipos de entes (entidades de existência e mobilidade): elementar e composto [Barbosa and Geyer, 2001]. 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. Um ente elementar possui um comportamento - que implementa sua funcionalidade; e uma história - espaço de armazenamento e de comunicação/sincronização entre entes, que funciona como uma memória distribuída logicamente compartilhada. A história fica encapsulada no ente e, no caso dos entes compostos, é compartilhada pelos entes filhos. Os entes acessam somente a sua história e a do seu pai. Para se comunicar com outros entes, o ente deve migrar para dentro do ente destino da comunicação, tornando-se seu filho, e interagir através da história. Um fragmento de código dos entes `holo` e `contrib` da aplicação Reuni@o é ilustrado na Figura 6.

A Hololinguagem foi alterada sintática e semanticamente para refletir os conceitos de adaptação ao contexto requeridos pela arquitetura ISAM, gerando a linguagem ISAMadapt. Nesta, o ente é uma unidade de modelagem e é definido como um **Objeto Pervasivo** (*PvO - pervasive object*) cujas características são:

- tem mobilidade e distribuição implícita;
- tem uma estrutura de composição expressa na relação de pai-fiho (árvore);
- migra por motivo de comunicação ou proximidade de recursos;
- seu mecanismo de comunicação é baseado no espaço de tuplas que reflete um desacoplamento temporal e espacial (comunicação anônima e assíncrona) o qual permite o tratamento da desconexão;
- durante a execução, seu código é carregado da ISAMbda sob demanda, e seus dados são armazenados em um espaço de tuplas externo a ele;
- seus dados persistentes são armazenados no Ambiente Virtual do Usuário;
- se adaptativo, existe um objeto-adaptador associado a ele, o qual conterá o código a ser executado correspondente ao estado do elemento de contexto corrente.

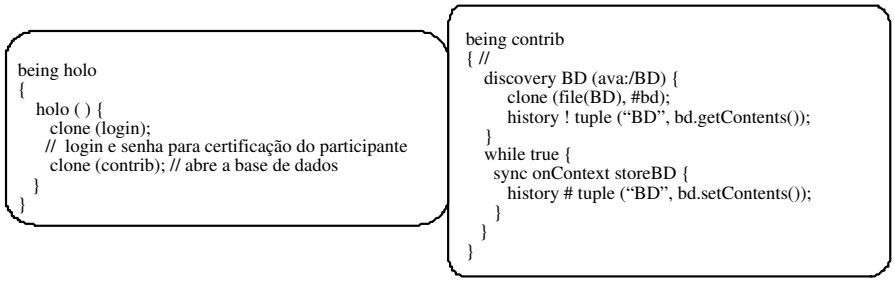


Figura 6: Fragmento de código de um ente

4.4. Codificando a Adaptação ao Contexto

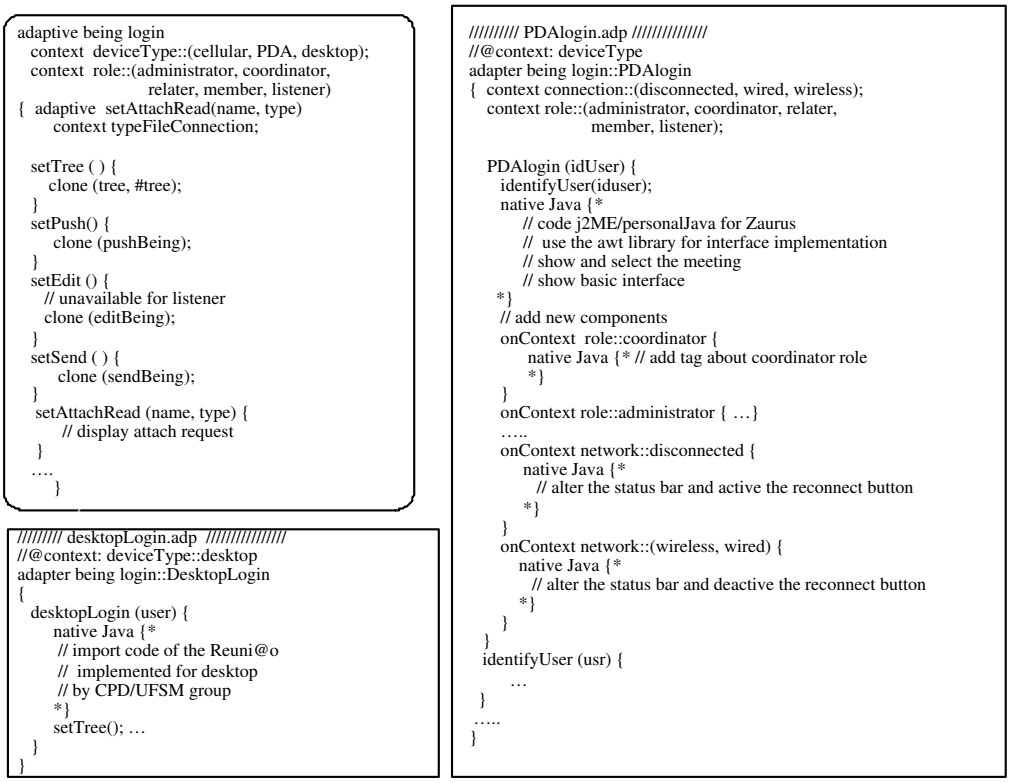
A adaptação ISAM requer a existência de múltiplos caminhos de execução da aplicação, ou configurações alternativas, as quais exibem diferentes níveis de utilização de recursos (elementos de contexto). Códigos alternativos são implementados através do conceito de **adaptadores**, introduzidos na linguagem. Este fornece o suporte para expressar a disponibilidade de alternativas de execução, e o ambiente de execução da linguagem fornece os meios para que o progresso da aplicação seja monitorado e influenciado pelo contexto (chaveamento para uma alternativa). Desta forma, a máquina de execução do middleware, `ISAMadaptEngine`, em tempo de execução, escolhe entre os códigos alternativos o mais adequado ao estado do elemento de contexto corrente, após receber a notificação de alteração no contexto pelo `ISAMcontextService` [Augustin, 2004].

No código, as estratégias de adaptação podem ser implementadas de diferentes formas: (a) usando métodos e entes adaptativos, que implementam adaptação de código; (b) usando comandos contextualizados, que implementam adaptação paramétrica; (c) definindo políticas de adaptação.

4.4.1. Adaptação Usando Entes e Métodos

ISAMadapt introduziu os conceitos de ente e método adaptativo. **Ente adaptativo** (`adaptive being`) é aquele cujo código é determinado pelo estado do elemento de contexto. Em geral, está associado ao código de carga, na instanciação do ente (comando `clone`). Por exemplo, o ente `login` (Figura 7) cujo código é dependente do tipo de dispositivo que dispara sua criação. Se nem todo o ente é adaptativo, mas somente métodos deste, pode-se usar o conceito de **método adaptativo**, método cuja funcionalidade é adaptativa ao contexto. Estes são identificados no código pelo qualificador `adaptive` acrescido ao nome do método chamado. Por exemplo, a declaração de método adaptativo `expandTree` na Figura 8, que é sensível à variação do estado do elemento de contexto `connection`.

A implementação dos códigos alternativos para o ente adaptativo é através de adaptadores (`adapters`), os quais modelam comportamento aos estados do elemento de contexto considerado. Adaptadores são codificados de forma semelhante aos entes, porém em arquivos-fontes de extensão `adp`. Desta forma, adaptadores são construídos como pares [estado do ambiente em execução, procedimentos da aplicação para o estado]. A associação entre código e estado do elemento de contexto é realizada através do menu `Adapter` do `ISAMadapt IDE`, que gera o arquivo `adapters.xml` utilizado pela `ISAMadaptEngine` em tempo de execução. A Figura 7 esboça o fragmento de código correspondente aos adaptadores do elemento de contexto `deviceType` para o ente adaptativo `login`. Na Figura 8 são representados os adaptadores correspondentes ao método `expandTree` e `displayTree`.

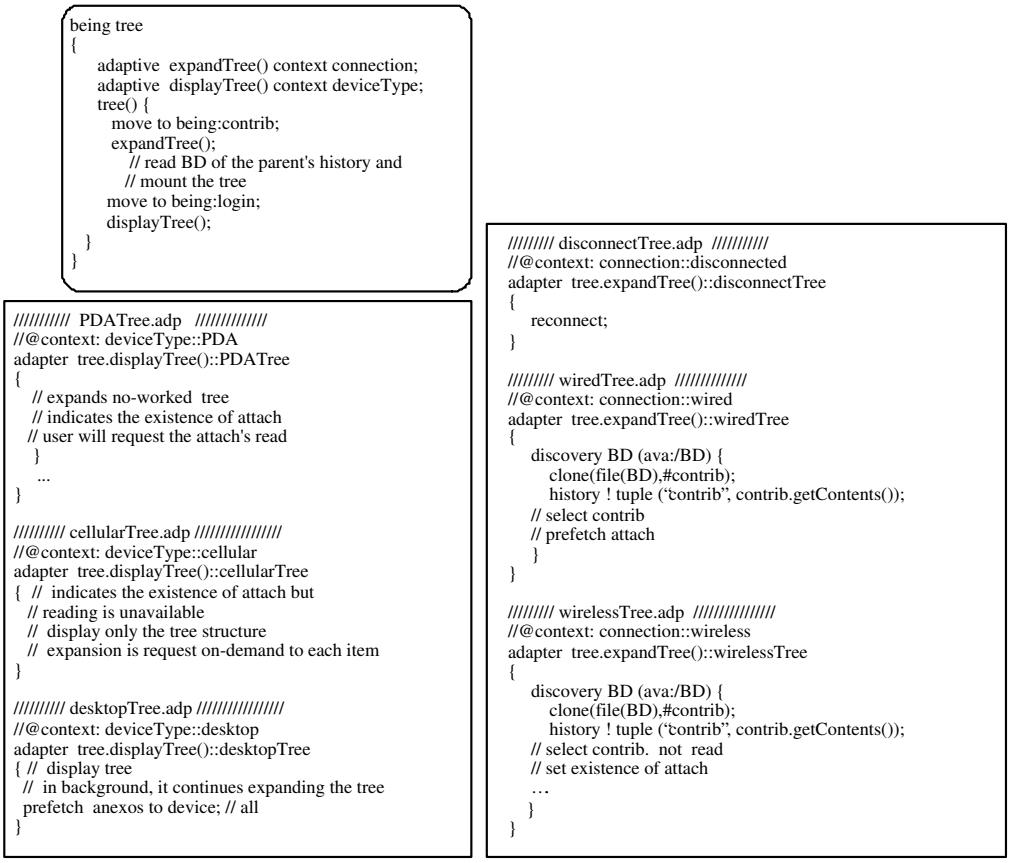


```
adaptive being login
   context  deviceType::(cellular, PDA, desktop);
   context  role::(administrator, coordinator,
                        relater, member, listener)
{  adaptive  setAttachRead(name, type)
       context typeFileConnection;

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

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

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

Figura 7: Fragmento de código de adaptadores de ente



```
being tree
{
     adaptive  expandTree() context connection;
     adaptive  displayTree() context deviceType;
     tree() {
        move to being:contrib;
        expandTree();
           // read BD of the parent's history and
           // mount the tree
       move to being:login;
       displayTree();
    }
}
```

```
//////////  PDATree.adp   ////////////////
//@context: deviceType::PDA
adapter  tree.displayTree()::PDATree
{
   // expands no-worked  tree
   // indicates the existence of attach
  // user will request the attach's read
   }
    ...
}

////////// cellularTree.adp /////////////////
//@context: deviceType::cellular
adapter  tree.displayTree()::cellularTree
{  //  indicates the existence of attach but
   // reading is unavailable
   //  display only the tree structure
   //  expansion is request on-demand to each item
}

////////// desktopTree.adp /////////////////
//@context: deviceType::desktop
adapter  tree.displayTree()::desktopTree
{ //  display tree
 //  in background, it continues expanding the tree
 prefetch  anexos to device; // all
}
```

```
//////// disconnectTree.adp  ///////////
//@context: connection::disconnected
adapter  tree.expandTree()::disconnectTree
{
    reconnect;
}

//////// wiredTree.adp  ////////////////
//@context: connection::wired
adapter  tree.expandTree()::wiredTree
{
    discovery BD (ava:/BD) {
       clone(file(BD),#contrib);
       history ! tuple (“contrib”, contrib.getContents());
    // select contrib
    // prefetch attach
     }
}

//////// wirelessTree.adp  ////////////////
//@context: connection::wireless
adapter  tree.expandTree()::wirelessTree
{
    discovery BD (ava:/BD) {
       clone(file(BD),#contrib);
       history ! tuple (“contrib”, contrib.getContents());
    // select contrib.  not  read
    // set existence of attach
    …
    }
}
```

Figura 8: Fragmento de código de adaptadores de método

4.4.2. Adaptação Usando Comandos Contextualizados

As construções baseadas em contexto são: (i) Declaração do elemento de contexto ao qual o ente é sensível. Por exemplo `context network::(connected, disconnecting)` indica que a aplicação se adapta ao estado conectado/desconectando da rede. Esta declaração permite o uso do contexto em comandos de seleção e repetição. Por exemplo,

```
if network::disconnecting
   then prefetch anexos to device;
```

(ii) Comando `onContext`. Este comando indica que o bloco de comandos internos a ele deverá ser executado quando o estado do elemento de contexto corresponder ao definido. A seqüência de execução segue no próximo comando, exceto se for especificado o qualificador `sync`, tornando o comando síncrono. Por exemplo,

```
onContext role::coordinator {
      native Java {* // instance tag Users
*} ...
```

indica que se o papel do participante é de coordenador a tag correspondente deve ser instanciada na interface deste usuário. O código nativo em Java permite o uso de pacotes Java não disponíveis na linguagem-base, como Swing e AWT para interfaces gráficas.

(iii) ISAMadapt fornece comandos de adaptação para modelar estratégias comuns nas aplicações móveis adaptativas. 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 definidas pelo programador da aplicação [Yamin, 2004]. Os comandos fornecidos são: `move` (migração), `clone` (criação de entes), `prefetch` (cópia local de arquivos), `push` (envio de dados), `disconnect` (desconexão lógica e planejada), `reconnect` (reconexão), `install` (instalação de aplicações no AVU), `reschedule` (força uma reavaliação do escalonamento), `discovery` (descoberta de recursos e serviços). Entes de sistema, como `file` e `filter`, também têm um comportamento adaptativo ao contexto [Augustin, 2004]. Um exemplo é dado no fragmento de código da Figura 8, ente `tree`, o comando `move to being:login` indica que o ente `tree` deverá se mover para dentro do ente `login`, após capturar as informações da base de contribuições, para que a árvore possa ser expandida na interface do usuário.

4.4.3. Politicas de Adaptação

Em geral, a aplicação não necessita interferir em decisões de gerenciamento da execução executadas pelo middleware, como por exemplo: decidir em qual nodo o objeto *pervasivo* deverá ser criado é atribuição do Serviço de Escalonamento Adaptativo do EXEHDA. Porém, a aplicação pode especificar políticas que definem: (a) estratégia de escalonamento a adotar, selecionada a partir do estado da conexão à rede; (b) procedimento em caso de desconexão (relativo ao comando `disconnect`), tal como enviar os entes para nodos da rede fixa; (c) âncoras para os entes, que podem ser outros entes ou recursos. Um exemplo de políticas de adaptação é apresentado na Figura 9 e indica que o ente `tree` deve ser criado (fisicamente) próximo ao recurso da base de dados das contribuições. O nome lógico `BD` será resolvido pelo serviço `Discoverer` do middleware, conforme descritor do recurso que se encontra no Ambiente Virtual da Aplicação. A outra política define que o ente `tree` é ancorado pelo ente `login`, ou seja, se o ente `login` migrar, o ente `tree` o acompanhará. O programador pode usar o menu `Policies` do

```
<policy composition= “replace”>
 <being name="tree”/>
 <closerTo type=”resource:BD”/>
     // tree is created closer to data base after resolution of logic name BD
 <anchor type="being:login”/>
     // if login being moving, the tree being follows him
 </policy>
```

Figura 9: Exemplo de política de adaptação

`ISAMadapt IDE` para criar o arquivo `policies.xml`, que será utilizado pelo middleware quando da instanciação dos serviços que atendem a aplicação.

5. Compilando e Executando a Aplicação

A estratégia usada para disponibilizar rapidamente a linguagem foi traduzi-la para a linguagem Java. Para isso, usou-se a gramática ISAMadapt como entrada para a ferramenta JavaCC, e obteve-se um tradutor ISAMadapt [Augustin, 2004]. Este é usado para receber como entrada o programa-base e os adaptadores, gerando o programa Java correspondente. O tradutor insere no código Java as chamadas necessárias ao `ISAMcontextService, ISAMadaptEngine` e aos outros serviços do middleware que dão suporte à linguagem [Yamin, 2004]. Esta solução permite-nos utilizar recursos de Java como *threads*, e carga dinâmica de código na implementação da linguagem ISAMadapt.

Após a tradução, o código Java gerado é empacotado e gerado o descritor da aplicação, com o auxílio do menu `Build` do `ISAMadapt IDE`. O descritor é um documento XML, o qual agrupa uma série de meta-dados que provêem uma descrição abstrata da aplicação ISAMadapt. Estes códigos são armazenados na `ISAMbda`, a qual disponibiliza o acesso *pervasivo* ao código.

O código do Reuni@o está sendo 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`. Após ser selecionada a aplicação, o Descritor de Disparo da Aplicação é carregado (`Reuniao.isam`), e o middleware passa a controlar a execução da aplicação, baseado nas políticas definidas para essa aplicação [Augustin, 2004].

6. Trabalhos Relacionados

Somente nesse início de década surgiram projetos que visam desenvolver a infra-estrutura de software para o ambiente *pervasivo*. Dois projetos se destacam: Gaia [Roman, 2003] e Aura [Garlan et al., 2002]. 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*). Gaia é um middleware experimental usado para prototipar o gerenciamento de recursos e fornecer uma interface orientada ao usuário. A pesquisa limita o espaço físico para o espaço usado pelos professores: classes de aula, escritórios e salas de leitura. 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 pelo programador da aplicação. ISAM inclui também uma linguagem, e permite desenvolver novas aplicações adequadas ao ambiente *pervasivo*, enquanto que Gaia visa a adaptação de aplicações existentes.

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 é correspondente ao Ambiente Virtual do Usuário, definido no ISAM. Aura é um grande projeto que investiga novas arquiteturas para o ambiente pervasivo focado na dimensão pessoal do usuário, suas tarefas e preferências. ISAM propõe 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*). 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 projeto ISAM, mas cujos componentes para implementação estão sendo construídos pelo middleware EXEHDA (instanciação remota, base de dados e código *pervasiva*, perfil do usuário, contexto).

Considerando especificamente a área de linguagens de programação, soluções para a computação *pervasiva* ainda não tinham sido abordadas até o início do projeto ISAM. Alguns trabalhos abordavam adaptação específica a um elemento, como a linguagem Klaim [Bettini et al., 2001] que permite construções para projetar o comportamento dinâmico da conectividade, ou C2 style [Bellavista and et al., 1999] para mobilidade de código.

A solução ISAMadapt, embora não tenha sido influenciada diretamente pelos conceitos da orientação a aspectos, retém semelhanças com esta. O programa-base da orientação a aspectos, pode ser associado ao programa escrito em ISAMadapt. Os programas-aspectos correspondem aos adaptadores, enquanto que os métodos e os entes adaptativos correspondem aos pontos de união [Kiczales and et al., 2001]. 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). ISAMadapt pode ser considerada uma linguagem-aspecto, porém com características dinâmicas, que envolvem uma associação com um middleware de execução.

7. Conclusões

A *Computação Pervasiva* é um novo paradigma que tem despertado muita atenção recentemente. A importância da adaptação neste ambiente é largamente reconhecida. Entretanto, adaptação dinâmica não é facilmente implementada. Hoje, as aplicações móveis que expressam comportamento adaptativo o fazem usando uma abordagem *ad-hoc*, específica da aplicação ou de uma classe de aplicações. Neste artigo, apresentou-se o ambiente ISAMadapt, que integra a arquitetura ISAM, e as abstrações da linguagem para expressar consciência do contexto. As abstrações para definir contexto, comportamento alternativo e políticas de adaptação, foram adicionadas à uma linguagem com mobilidade implícita e interação por multi-espaços de objetos. A semântica definida para essas abstrações procura contribuir para reduzir a complexidade do desenvolvimento de aplicações dinâmicas que seguem a semântica siga-me da *Computação Pervasiva*.

Referências

Augustin, I. (2004). *Abstrações para uma Linguagem de Programação Visando Aplicações Móveis em um Ambiente de Pervasive Computing*. Tese (doutorado), Instituto de Informática, UFRGS, Porto Alegre.

Augustin, I., Yamin, A., Barbosa, J., da Silva, L., Real, R., and Geyer, C. (2003). *Mobile Computing Handbook*, chapter ISAM, Joing Context-awareness and Mobility to Building Pervasive Applications. CRC Press, New York. (to appear).

Augustin, I., Yamin, A., Barbosa, J., and Geyer, C. (2002a). Isam - a software architecture for adaptive and distributed mobile applications. In *IEEE Symposium On Computers And Communications, ISCC*, page 7, Taormina, Italy. New York: IEEE Computer Society.

Augustin, I., Yamin, A., da Silva, L., Barbosa, J., and Geyer, C. (2002b). Towards taxonomy for mobile applications with adaptive behavior. In *International Symposium On Parallel And Distributed Computing And Networks, PDCN 2002*, Innsbruck, Austria. Calgary, Canada: Acta Press.

Barbosa, J. and Geyer, C. (2001). Uma linguagem multiparadigma orientada ao desenvolvimento de software distribuído. In *V Simpósio Brasileiro de Linguagens de Programação*.

Bellavista, P. and et al. (1999). A secure and open mobileagent programming environment. In *ISADS'99*. IEEE Press.

Bettini, L., Loreti, M., and Pugliese, R. (2001). Modeling node connectivity in dynamically evolving networks. In *CONCOORD International Workshop on Concurrency and Coordination, v.54 of ENTCS*.

Garlan, D., Steenkiste, P., and Schmerl, B. (2002). Project aura: Toward distraction-free pervasive computing. *IEEE Pervasive Computing*, 1(3). New York.

Kiczales, G. and et al. (2001). An overview of aspectj. In *European Conference On Object-Oriented Programming, ECOOP*, page 15, Budapest, Hungary. Berlin: Springer-Verlag.

Roman, M. (2003). *An Application Framework for Active Space Applications*. Ph.d. thesis, University of Illinois, Urbana-Champaign, Illinois, USA. `http://www.cs.uiuc.edu/gaia` [april 2003].

Saha, D. and Mukherjee, A. (2003). Pervasive computing: a paradigm for the 21st century. *IEEE Computer*, 36(3):25–31.

Satyanarayanan, M. (2001). Pervasive computing: Vision and challenges. *IEEE Personal Communications*, 4(8). New York.

Yamin, A., Augustin, I., Barbosa, J., da Silva, L., Real, R., Cavalheiro, G., and Geyer, C. (2003). Towards merging context-aware, mobile and grid computing. *Journal of High Performance Computing Applications*.

Yamin, A. C. (2004). *Arquitetura para um Ambiente de Execução Direcionado às Aplicações Móveis Conscientes do Contexto em um Ambiente de Pervasive Computing*. Proposta de tese (doutorado em ciência da computação), Instituto de Informática, UFRGS, Porto Alegre.

Yamin, A. C., Augustin, I., da Silva, L. C., Real, R. A., Barbosa, J., and Geyer, C. F. R. (2004). EXEHDA: an adaptive middleware for the pervasive computing scenery. WWW. `http://www.inf.ufrgs.br/~isam`.