Uma Arquitetura para Controle da Adaptação Dinâmica na Computação Ubíqua

Nelsi Warken[1,3], **Adenauer C. Yamin**[1,2]

[1]PPGINF – Universidade Católica de Pelotas (UCPEL)

[2]CDTec – Universidade Federal de Pelotas (UFPEL)

[3]ATI – Centro de Pesquisa Agropecuária Clima Temperado (EMBRAPA-CPACT)

Abstract. *The main purpose of this study is to promote the use of concepts and technologies for the Semantic Processing and Autonomics Systems in the design of mechanisms to control the adaptation to the context in Ubiquitous Computing. The research developed has as focus to concept a proposal to control the adaptations, considering the context, generated by tracked information, semantic information and inferences using these same information. The premisse is a generic mechanism for adaptation that can be used in run-time, both by middleware or by applications. The proposed model was evaluated by two case studies, presenting satisfactory results regarding the fulfillment of the demands of ubiquity.*

Resumo. *O objetivo central deste trabalho é promover o emprego dos conceitos e tecnologias referentes a Processamento Semântico e Sistemas Autônomos na concepção de mecanismos para controle da adaptação ao contexto na Computação Ubíqua. A pesquisa tem como eixo conceber uma proposta para controlar as adaptações, considerando o contexto produzido por informações monitoradas, informações semânticas e inferências a partir destas mesmas informações. A premissa é culminar em um modelo de adaptação genérico, que poderá ser utilizado tanto pelo middleware, quanto pelas aplicações. O modelo proposto foi avaliado por dois estudos de caso, apresentando resultados satisfatórios quanto ao atendimento das demandas de ubiquidade.*

1. Introdução

Na Computação Ubíqua (UbiComp) os sistemas precisam monitorar e se adaptar ao ambiente, compreendendo o contexto em que estão inseridas. Essa nova classe de sistemas computacionais, adaptativos ao contexto, abre perspectivas para o desenvolvimento de aplicações mais ricas, elaboradas e complexas, que exploram a natureza dinâmica das modernas infraestruturas computacionais e a mobilidade do usuário. Entretanto, o desenvolvimento de aplicações que se adaptem continuamente ao ambiente e permaneçam funcionando mesmo quando o indivíduo se movimentar ou trocar de dispositivo [Costa et al. 2008], continua um desafio de pesquisa em aberto.

A Computação Autônoma, também denominada Computação Autonômica, preceitua que os sistemas computacionais devem desempenhar funções automáticas de configuração, tratamento, otimização e proteção, substituindo a execução rotineira de tarefas complexas por políticas descritas em alto nível por usuários e administradores. Uma

política é uma representação, em uma forma padrão, de um comportamento desejado e, também, das restrições associadas a este comportamento. As políticas orientadas a objetivos representam uma forma de alto nível de especificação comportamental, definindo objetivos a serem perseguidos e, deixando a aplicação ou o *middleware* determinar as ações necessárias para alcançar estes objetivos. Por sua vez, políticas de funções de utilidade estendem as abordagens orientadas a objetivos, contribuindo para determinação do objetivo mais importante, de maior utilidade, em uma dada situação [Kephart and Das 2007].

Ontologia é definida como uma especificação formal e explícita de uma conceituação compartilhada [Fensel 2000]. Ontologia, além de ser um elemento importante no processamento semântico, se mostra um instrumento capaz de especificar entidades ou elementos de contexto como usuários, dispositivos, nodos, serviços, localização, políticas, regras, entre outros, atuando como um padrão para descrições, características, configurações e perfis, que poderão ser compartilhados pelas aplicações e serviços na Computação Ubíqua.

O modelo de controle da adaptação proposto denomina-se EXEHDA-DA (*Execution Environment for Highly Distributed Applications - Dynamic Adaptation*), e sua concepção considerou a premissa do mesmo vir a ser integrado no *middleware* EXEHDA [Yamin 2004].

O objetivo central do trabalho é avaliar a exploração das tecnologias de processamento semântico e Sistemas Autônomos como mecanismos de controle da adaptação ao contexto na computação ubíqua, considerando as principais características e desafios de pesquisa da mesma. Será criado um modelo ontológico para o ambiente computacional provido pelo *middleware* EXEHDA, e a proposta é tomar decisões automáticas de adaptação para este ambiente, com base em informações monitoradas, informações semânticas e inferências a partir das mesmas.

2. Modelo Semântico Proposto

Com o intuito de ampliar a flexibilidade na especificação das diferentes adaptações, o EXEHDA-DA suporta que sejam definidos requisitos por aplicação. A premissa é que vários elementos de contexto podem ser relevantes para uma mesma aplicação na UbiComp, e que a possibilidade de várias adaptações pode contribuir para manter e até mesmo melhorar a qualidade da execução quando o contexto muda.

O modelo semântico proposto para o domínio do EXEHDA-DA contempla a seguinte estrutura:

- **OntUbi** - Ontologia com as entidades (classes), seus atributos e relacionamentos do contexto de interesse das aplicações ubíquas. Na OntUbi os elementos de contexto ou entidades, chamados de Classes na OWL, possuem três categorias básicas de informações: (i) recursos de hardware; (ii) recursos de software; (iii) recursos de usuário. As seguintes ontologias também compõem a OntUbi:
  - **OntContext** - Ontologia da Situação do contexto: representa os contextos coletados, contextos notificados e os contextos de interesse da aplicação.
  - **OntAdapt** - Ontologia da Política de Adaptação da Aplicação: regras, parâmetros, operações e preferências, restrições e ações de adaptação para os componentes das aplicações.

– **OntHistAdapt** - Ontologia prevista para o registro das decisões de Adaptação, histórico das adaptações realizadas.

De modo mais específico, a Política de Adaptação da Aplicação especificada na OntAdapt, tem a finalidade de registrar os perfis dos componentes das aplicações para a adaptação. É composta de especificações de regras que determinam o comportamento dos componentes da aplicação. As classes `Aplicacao`, `Componente`, `Adaptador`, `Operacao`, `Param_Tipo`, `Param_Valor`, `Sensor` e `Adapt` são subclasses da superclasse `Software`. Na OntAdapt podem ser definidas políticas para várias aplicações. Cada aplicação é composta por diversos componentes, instâncias do relacionamento `Aplicacao_Componente`. Cada componente pode ter várias adaptações, instâncias do relacionamento `Componente_Adaptador`. Cada adaptação de cada componente da aplicação pode ter vários parâmetros, instâncias do relacionamento `Adaptador_ParamTipo`. Um Parâmetro pode ter várias ocorrências de valor inferior, valor superior e valor de utilidade, instâncias do relacionamento `ParamTipo_ParamValor`. Esta ontologia é instanciada em tempo de desenvolvimento da aplicação e utilizada em tempo de execução, pelo serviço de controle de adaptação dinâmica, para orientar as adaptações ao contexto dos componentes. O modelo ontológico previsto permite uma evolução incremental das instâncias da OntAdapt, tais como regras, parâmetros, adaptadores e operações [Warken 2010].

3. EXEHDA-DA: Visão Geral e Modelagem

O modelo de controle da adaptação dinâmica de aplicações em ambiente ubíquo, quando da tomada de decisão de adaptação, considera as categorias de informações: (i) o contexto, com base em informações monitoradas, informações semânticas e inferências a partir destas mesmas informações; (ii) política de adaptação da aplicação; (iii) preferências do usuário. Para isto, o EXEHDA-DA comunica-se, através de interfaces definidas, tanto com as aplicações pelos comandos de adaptação, quanto com os outros serviços do *middleware* empregando notificações, dados contextuais e decisões de adaptação, facultando que o mesmo possa ser desenvolvido, modificado, e estendido de forma independente.

O EXEHDA-DA contempla um modelo de adaptação Multi-nível Colaborativa organizado em dois níveis [Warken 2010]. No nível da aplicação, em tempo de desenvolvimento são previstas duas definições: (i) criação das ontologias da Política de Adaptação da Aplicação e Contexto de Interesse da Aplicação e; (ii) programação dos comandos adaptativos nos códigos dos componentes de software das aplicações. Ainda no nível da aplicação, em tempo de execução, são ativados os comandos adaptativos que acionam serviços do *middleware* EXEHDA. Por sua vez no nível do serviço (*middleware*) acontece a decisão de adaptação, considerando o estado do contexto, a Política de Adaptação da Aplicação e as preferências do usuário.

O EXEHDA-DA considera dois tipos de adaptação dinâmica [Warken 2010]:

- Adaptação Não-Funcional: consiste na capacidade do sistema atuar sobre a localização física dos componentes das aplicações, seja no momento de uma instanciação do componente, seja, posteriormente, via migração do mesmo. Atua sobre a gerência da execução distribuída, através de operações de mapeamento, reescalonamento, instanciação remota e migração;

- Adaptação Funcional: consiste na capacidade do sistema atuar sobre a seleção da implementação do componente de software a ser utilizada em um determinado contexto de execução.

O EXEHDA-DA comunica-se, através de interfaces definidas, tanto com as aplicações, quanto com os outros serviços do *middleware*, utilizando notificações, dados contextuais e decisões de adaptação, facultando que o mesmo possa ser desenvolvido, modificado, e estendido de forma independente.

A figura 1 apresenta a arquitetura de software do Serviço de Adaptação Dinâmica ao Contexto, EXEHDA-DA, com seus módulos, as interfaces entre os mesmos, e a comunicação com os outros serviços do *middleware*. Uma descrição da mesma é feita a seguir:

- **Tratamento do Contexto Notificado**: recebe do Serviço de Reconhecimento de Contexto, EXEHDA-SS, a identificação do contexto notificado. Acessando as informações de mudança de contexto de interesse do componente da aplicação, as informações computacionais do contexto para a adaptação, valores dos sensores do ambiente computacional. O mecanismo de tratamento de contexto notificado também consulta a política da aplicação, através do processamento semântico, e acessa as informações necessárias para o cálculo da regra de adaptação em questão. Estas informações são os tipos, valores e utilidades de parâmetros necessários para o cálculo da regra de adaptação, configurados na OntAdapt para a adaptação do componente da aplicação. Neste módulo são preparadas todas as informações necessárias para a tomada de decisão da adaptação.
- **Processamento da Regra do Adaptador**: este processamento produz uma lista com uma ou mais ações de adaptação, classificada por critério de utilidade. Através da API Jena [Warken 2010], são inferidas as possíveis opções para ação adaptativa, utilizando a regra de adaptação e os valores de parâmetros necessários e oferecidos para calculá-la. Utilizando a função de utilidade serão classificadas as possíveis ações, em ordem de mais alta utilidade de ação adaptativa para a mais baixa. A adaptação funcional e a adaptação não-funcional são computadas da mesma maneira, através de regra de adaptação.
- **Processamento da Regra do Usuário**: são acessadas as preferências do usuário para a Aplicação na classe Usuario da OntUbi. Através destas preferências classificadas por valor de utilidade, será selecionada, dentre as possíveis ações adaptativas, inferidas no módulo anterior, aquela que tiver maior valor de utilidade para o usuário, fornecendo-lhe um maior nível de satisfação. Se não for configurada a preferência do usuário para este tipo de adaptador e aplicação, será selecionada a ação de maior valor de utilidade inferida no mecanismo anterior.
- **Disponibilização da Adaptação Inferida**: neste módulo são retidas as informações necessárias para a ação de adaptação, inferida no módulo anterior. No momento da execução da adaptação, solicitada pelo comando adaptativo que está inserido no código do componente da aplicação em execução, o serviço Executor irá acessar as informações para execução da adaptação inferida.
- **Processamento Semântico**: neste módulo é utilizada a linguagem SPARQL, e a API Jena para acessar, instanciar e inferir informações necessárias para computar as regras de adaptação. Os produtos destas regras são entregues para o mecanismo

que faz a retenção das decisões de adaptação ao contexto para os componentes das aplicações. Estas informações estão nas ontologias utilizadas neste serviço: OntUbi, OntContext e OntAdapt.

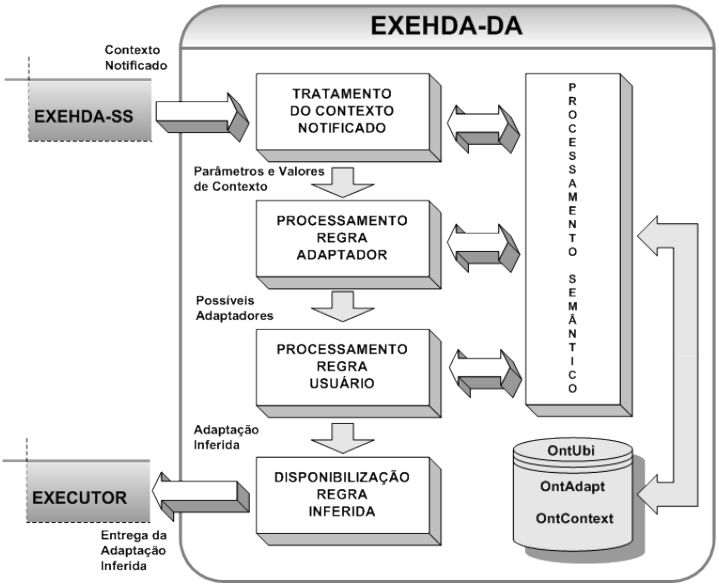

Figura 1. Arquitetura de Software do EXEHDA-DA

4. Estudos de Caso

O modelo de controle de adaptação do EXEHDA-DA por ser programável por regras personalizáveis por aplicação, permite a sua utilização para vários tipos de cenários de uso, em diferentes domínios de contexto. Neste sentido, foram desenvolvidos dois estudos de caso, um explorando o controle da adaptação funcional e outro o controle da adaptação não-funcional. A discussão completa destes estudos de caso encontra-se em [Warken 2010].

4.1. Acompanhamento Ubíquo de Pacientes - AUP

A premissa buscada é qualificar o acompanhamento de pacientes, que não estejam internados em Unidades de Tratamento Intensivo (UTI). Seus objetivos são: (i) exibir dados de pacientes adquiridos dinamicamente por mecanismo de sensoreamento de sinais; (ii) emitir, de forma automatizada, diferentes níveis de alertas, em função dos dados sensoreados, para os agentes de saúde; (iii) integrar o serviço de alertas da aplicação a rede aberta de comunicação *Google Talk*; (iv) prover possibilidade de ter acesso, tanto a partir de dispositivo móveis, como de mesa; (v) permitir acesso ubíquo ao histórico dos dados sensoreados dos pacientes por agentes de saúde.

A AUP foi concebida para explorar adaptações funcionais de duas naturezas. Estas adaptações são especificadas na OntAdapt e atendem duas situações:

1. Adaptação funcional em função dos dados de contexto sensoreados do paciente (sinais vitais), determinando o componente de software empregado para exibição do nível de alerta;

2. Adaptação funcional em função do dispositivo que está sendo utilizado pelo usuário, selecionando o componente com a interface mais adequada ao dispositivo.

Os dispositivos tratados no processo adaptativo são do tipo desktop e PDA. Quando o enfermeiro desloca-se no hospital carrega o PDA. Por outro lado, tanto o enfermeiro como o médico utilizam um desktop e/ou notebook, no consultório ou no plantão do hospital. O PDA utilizado foi o Sharp Zaurus 5600 com resolução de 320x240 pixels, por sua vez, o desktop contemplou uma resolução mínima de 800x600 pixels.

4.2. Alocação Dinâmica de Recursos - ADR

O objetivo central da ADR é explorar de modo oportunista os recursos computacionais em uma infraestrutura computacional distribuída. Nesta perspectiva, as tarefas da aplicação serão escalonadas a medida que os recursos se tornarem disponíveis.

O ambiente computacional de testes da ADR contempla quatro tipos de software:

- **aplicação paralela CalcPi** - calcula o número π pelo método Monte Carlo;
- **aplicação CpuSteal** - atua como um gerador de cargas, operando em ciclos de ativação e desativação;
- **aplicação LoadGen** - produz uma descrição de carga computacional. São gerados tempos de ativação e desativação de processador, baseado em uma distribuição de probabilidades exponencial, para ser utilizada pelo CpuSteal;
- **aplicação MemSteal** - promove a ocupação de memória. Se vale da instanciação de matrizes para gerar demanda de memória nos nodos processadores.

As premissas de teste da ADR consideradas são: (i) a aplicação, cujas computações serão escalonadas, é do tipo sintético, e tem por objetivo central permitir a exploração dos recursos do EXEHDA-DA quando do controle das **adaptações não-funcionais**; (ii) duas arquiteturas paralelas do tipo *Cluster* constituem a grade computacional da infraestrutura de testes.

O critério geral é disponibilizar para a aplicação CalcPi, os nodos com maior poder computacional e menor nível de ocupação de processador e memória. Inicialmente serão alocados os nodos do *cluster* de preferência do usuário, uma vez esgotados os mesmos, serão escalonados os nodos melhor qualificados de outro *cluster*.

5. Trabalhos Relacionados

O estudo de trabalhos relacionados envolveu a avaliação de 7 trabalhos, cujos escopos contemplavam similaridade com o EXEHDA-DA. A tabela 1 apresenta uma comparação entre estes trabalhos, tendo por base características consideradas na modelagem do EXEHDA-DA. Os campos marcados com "+" indicam que o modelo possui a característica. Os campos sem marcação indicam que a ferramenta não possui a funcionalidade especificada. As características consideradas na comparação dos trabalhos relacionados com o EXEHDA-DA, foram:

- 01 - Adaptação funcional da aplicação e/ou do middleware;
- 02 - Adaptação não-funcional;
- 03 - Controle da adaptação externo à aplicação;
- 04 - Modelagem Semântica para a política da adaptação da aplicação;

- 05 - Dispositivos Móveis;
- 06 - Reutilização de políticas a partir de um catálogo;
- 07 - Tratamento autonômico da adaptação baseado em regras;
- 08 - Função de Utilidade.

Tabela 1. Comparativo dos Trabalhos relacionados ao modelo EXEHDA-DA.

Comparações / Modelos	01	02	03	04	05	06	07	EXEHDA-DA
Adaptação Funcional	+		+	+	+	+	+	+
Adaptação Não-funcional		+			+			+
Controle adaptação externo	+	+	+	+	+	+	+	+
Modelagem Semântica						+		+
Dispositivos Móveis	+				+		+	+
Reutilização de Políticas	+		+	+				+
Baseado em Regras	+	+		+	+	+	+	+
Função de Utilidade	+				+			+

- 01. **Carisma** [Capra et al. 2003], sistema baseado em um *Middleware* Reflexivo *Context-aware* para Aplicações Móveis. Embora as políticas baseadas em regras adotadas no Carisma se mostrem simples de utilizar, existem algumas desvantagens em comparação com as funções de utilidade do EXEHDA-DA. Em primeiro lugar, as regras prevêem menor transparência para o desenvolvedor, exigindo raciocínio em termos de ações de reconfiguração de baixo nível, em vez de um projeto arquitetural a nível semântico. No EXEHDA-DA, ações de reconfiguração de nível mais baixo são automaticamente determinadas pelo *middleware* e a adaptação é conduzida por atributos para adaptações funcionais e não-funcionais, enquanto as políticas baseadas em regras do Carisma não consideram previsão de adaptações não-funcionais. Ainda, Carisma manuseia apenas um número fixo de políticas para uma determinada adaptação, enquanto a abordagem do EXEHDA-DA com a função de utilidade é possível selecionar a melhor adaptação entre as alternativas, que são inferidas pelo servidor de adaptação.
- 02. **Chisel** [Keeney and Cahill 2003], é um *framework* para adaptações dinâmicas de serviços utilizando reflexão controlada por políticas, de maneira consciente ao contexto. É baseado na decomposição de aspectos extra-funcionais (não-funcionais) de uma aplicação em comportamentos alternativos. Em Chisel as políticas são definidas com uma linguagem declarativa e essencialmente baseadas em regras, realizando adaptações não-funcionais. O EXEHDA-DA, além de regras, se vale de funções de utilidade no cômputo das decisões de adaptação, assim como, contempla adaptações funcionais e não-funcionais.
- 03. **QUO** (Objetos de Qualidade) [Heineman et al. 2004, Sharma et al. 2004], é um projeto de *middleware*, que embora não trate todos os aspectos associados a dispositivos portáteis e redes sem fio, teve um impacto importante entre os sistemas adaptativos e projetos de desenvolvimento de *middleware QoS-aware*. Uma desvantagem da arquitetura Quo é que ela só se concentra na adaptação dos aspectos funcionais da aplicação, em função do estado dos recursos do sistema, não

considera adaptações não-funcionais quando do disparo de uma aplicação distribuída, bem como não tem suporte para dispositivos portáteis e redes sem fio, ao contrário do EXEHDA-DA. QUO, como o EXEHDA-DA, é também um dos poucos projetos que tentou abordar adaptação como um componente ou serviço reutilizáveis.

- 04. **Rainbow** [Garlan et al. 2004], consiste em um *framework*, uma linguagem e um processo incremental de engenharia de auto-adaptação. As técnicas de adaptação propostas pela abordagem Rainbow são semelhantes ao EXEHDA-DA, na medida em que separa a adaptação das funções lógicas da aplicação. No entanto, o Rainbow não foca nas exigências pertinentes aos ambientes com dispositivos móveis. Além disso, os seus desenvolvedores basearam as estratégias de adaptação em regras situação-ação, que especificam exatamente o que fazer em determinadas situações. Por sua vez, EXEHDA-DA usa políticas de objetivos estendidas expressas como funções de utilidade, que é um nível mais alto de especificação das estratégias de adaptação, definindo objetivos e implementando essas políticas, através de regras lógicas.
- 05. **Madam** [Geihs et al. 2009], é um projeto europeu de pesquisa com parceiros distribuídos em indústrias e universidades. Madam, além de um *middleware*, contempla uma metodologia de desenvolvimento *model-driven*, que se baseia em modelos de adaptação e as correspondentes transformações modelo-para-código. EXEHDA-DA não tem foco em ferramentas de desenvolvimento de software, seu objetivo é prover suporte para adaptações que possam ser utilizadas por componentes das aplicações. Madam não utiliza modelagem semântica para política de adaptação da aplicação, assim como não explicita se as políticas de adaptação podem ser reutilizadas para novas adaptações ou para outros componentes de software.
- 06. **Proteus** [Toninelli et al. 2007], emprega um modelo semântico na gerência da adaptação dos acessos permitidos aos usuários. Proteus realiza adaptações nas políticas de acesso à recursos, utilizadas pelas aplicações. Na gerência do processo adaptativo, de forma análoga ao EXEHDA-DA, contempla o uso de um modelo semântico. Por sua vez, o EXEHDA-DA prevê um espectro mais amplo de adaptações, estando sujeitas ao processo adaptativo diferentes funcionalidades de aplicações com naturezas diversas.
- 07. **SECAS** [Chaari and Laforest 2007], *Simple Environment for Context Aware Systems*, é um projeto que trata com a adaptação das aplicações ao contexto, considerando as preferências do usuário, do ambiente e os dispositivos envolvidos. O projeto SECAS preocupa-se apenas com as adaptações funcionais para aplicações da área médica. Utiliza expressões lógicas para as configurações de contexto para os adaptadores. Diferentemente do EXEHDA-DA, não utiliza a modelagem semântica para as regras de adaptação e sim um formalismo baseado em Redes de Petri.

De modo geral, pode-se dizer que funções de utilidade proporcionam uma maior uma maior possibilidade de adequação às necessidades do usuário, quando da adaptação. Mas é uma facilidade que não é considerada na maioria dos projetos.
Além disso, o EXEHDA-DA permite, a qualquer momento, o desenvolvedor da aplicação incluir novas instâncias nas classes da OntAdapt, como novos adaptadores, novas regras,

novos parâmetros, devido à política da aplicação ser mantida externamente, qualificando a configuração dos perfis das aplicações, através das regras, parâmetros e operações de sua política de adaptação.

6. Conclusão

A partir da avaliação e comparação dos trabalhos relacionados, é possível concluir que as funções de utilidade proporcionam um melhor ajuste da adaptação e, portanto, uma maior adequação às necessidades do usuário, porém este aspecto não vem sendo considerado na maioria dos projetos.

No EXEHDA-DA, devido a Política de Adaptação da Aplicação ser mantida externamente ao código da aplicação, é facilitado ao desenvolvedor incluir novas adaptações, novos contextos de interesse, novas regras e parâmetros operacionais. Neste sentido é disponibilizado um *framework* (FWADAPT) como interface para instanciação desta Política da Adaptação.

Por outro lado, o formalismo, a expressividade, a possibilidade de relacionamentos dinâmicos e inferências entre as informações, introduzidos pelo modelo semântico, facultam a definição de um modelo de Política de Adaptação em alto nível, bem como potencializam a manutenção, a reutilização, a padronização e o compartilhamento das informações do modelo ontológico entre os diversos serviços do *middleware*.

Todo contexto mensurável pode ser utilizado para a tomada de decisões de adaptação, as quais podem explorar processamento semântico e inferências lógicas.

O modelo de adaptação proposto, pode ser utilizado em tempo de execução pelas aplicações e pelo próprio *middleware*, suportando tanto adaptações funcionais como não-funcionais. Diferentemente da maioria dos outros modelos, que trabalham apenas com um tipo de adaptação dinâmica.

A gerência da decisão de adaptação é pró-ativa, podendo ser disparada a qualquer momento, e sem intervenção do usuário, em reação às mudanças de contexto. Este aspecto tem especial significado quando do tratamento de procedimentos emergenciais.

Diversas atividades estão em andamento, dentre estas destacaríamos a implementação de regras de adaptação que modifiquem parâmetros de outras regras, caracterizando uma adaptação ao contexto do próprio EXEHDA-DA.

Também está sendo avaliada a expansão das funcionalidades do *framework* FWADAPT com o objetivo de promover (i) a utilização da Classe Operacao na composição da regra lógica do adaptador; (ii) a conversão automática da regra lógica em uma regra que possa ser usada diretamente pelo EXEHDA-DA através da API Jena/SPARQL e; (iii) mostrar a regra lógica num formato mais amigável para o usuário.

No tocante a evolução do modelo de Controle da Adaptação Dinâmica ao Contexto do EXEHDA-DA, está prevista uma revisão tanto das taxonomias na área de aplicações ubíquas, bem como do modelo ontológico utilizado, sistematizando as funcionalidades necessárias para atendimento das novas demandas. Um aspecto a ser destacado é considerar na tomada de decisão adaptativa o histórico da mudança de contexto e o histórico de adaptações realizadas.

Em relação ao processo de implantação do EXEHDA-DA em um ambiente com

demandas potenciais, está sendo considerada a infraestrutura de pesquisa da Embrapa Clima Temperado, promovendo o uso do modelo de Controle da Adaptação Dinâmica ao Contexto em necessidades da Agropecuária de Precisão.

Referências

Capra, L., Emmerich, W., and Mascolo, C. (2003). Carisma: Context-aware reflective middleware system for mobile applications. *IEEE Transactions on Software Engineering*, 29(10):929–945.

Chaari, T. and Laforest, F. (2007). Adaptation in context-aware pervasive information systems: the secas project. *International Journal of pervasive Computing and Communications*, 3(4):400–425.

Costa, C. A., Yamin, A. C., and Geyer, C. F. R. (2008). Toward a general software infrastructure for ubiquitous computing. *IEEE Pervasive Computing*, 7(1):64–73.

Fensel, D. (2000). Ontologies-silver bullet for knowledge management and electronic commerce.

Garlan, D., Cheng, S.-W., Huang, A.-C., Schmerl, B., and Steenkiste, P. (2004). Rainbow: Architecture-based self-adaptation with reusable infrastructure. *Computer*, 37(10):46–54.

Geihs, K., Barone, P., Eliassen, F., Floch, J., Fricke, R., Gjørven, E., Hallsteinsen, S., Horn, G., Khan, M. U., Mamelli, A., Papadopoulos, G. A., Paspallis, N., Reichle, R., and Stav, E. (2009). A comprehensive solution for application-level adaptation. *Software Practice & Experience*, 39(4):385–422.

Heineman, G. T., Loyall, J. P., and Schantz, R. E. (2004). *Component Technology and QoS Management*, volume 3054 of *Lecture Notes in Computer Science*. Springer.

Keeney, J. and Cahill, V. (2003). *Chisel: A Policy-Driven, Context-Aware, Dynamic Adaptation Framework*. IEEE Computer Society.

Kephart, J. O. and Das, R. (2007). Achieving self-management via utility functions. *IEEE Internet Computing*, 11(1):40–48.

Sharma, P. K., Loyall, J. P., Heineman, G. T., Schantz, R. E., Shapiro, R., and Duzan, G. (2004). *Component-Based Dynamic QoS Adaptations in Distributed Real-Time and Embedded Systems*, volume 3291 of *Lecture Notes in Computer Science*. Springer.

Toninelli, A., Montanari, R., Kagal, L., and Lassila, O. (2007). Proteus: A semantic context-aware adaptive policy model.

Warken, N. (2010). Uma proposta de controle da adaptação dinâmica ao contexto na computação ubíqua. Tese de mestrado em ciência da computação, PPGINF/Centro Politécnico/UCPEL, Pelotas-RS. Disponível: http://olaria.ucpel.tche.br/nelsiw/.

Yamin, A. C. (2004). Arquitetura para um ambiente de grade computacional direcionada às aplicações distribuídas, móveis e conscientes de contexto da computação pervasiva. Tese de doutorado em ciência da computação, Instituto de Informática/UFRGS, Porto Alegre-RS.