Adaptação Dinâmica ao Contexto na Computação em Grade

Nelsi Warken
Embrapa Clima Temperado
Área de Tecnologia da Informação
BR 392 - KM 78, Pelotas/RS, Brasil
nelsi.warken@gmail.com

Adenauer Yamin
Universidade Católica de Pelotas
UCPEL - PPGINF
Universidade Federal de Pelotas
UFPEL - CDTec
adenauer@gmail.com

Cláudio Geyer
Universidade Federal do Rio Grande do Sul
UFRGS - II - PPGC
claudio.geyer@gmail.com

Abstract

The research presented in this paper has focus in the design of a mechanism for controlling the adaptations to the context of component-based applications in the Grid Computing, considering the context produced by monitored informations, semantic informations and inferences from these same informations. The premise is to culminate in a generic model of adaptation, which can be used by both, middleware and applications. To achieve this, we propose the use of the concepts and technologies related to the Semantic Processing and Autonomous Systems in the design of a mechanism to control the adaptation. The proposed model was assessed by case study, which characterizes the use of the mechanism for the Control of the Dynamic Adaptation as support for scheduling in computational grids, where resources are heterogeneous and with dynamic occupancy rate.

1. Introdução

Na Computação em Grade, as aplicações para usufruírem do caráter dinâmico da infraestrutura computacional oferecida, as mesmas precisam monitorar e se adaptar ao ambiente, compreendendo o contexto em que estão inseridas. Essa nova classe de aplicações, adaptativas ao contexto, abre perspectivas para o desenvolvimento de sistemas mais ricos, elaborados e complexos, que exploram a natureza dinâmica das modernas infraestruturas computacionais. Entretanto, o desenvolvimento de aplicações que se adaptem continuamente ao ambiente computacional disponível e permaneçam funcionando [3, 13], continua um desafio de pesquisa em aberto.

Por sua vez a Computação Autônoma preceitua que os sistemas computacionais devem desempenhar funções automáticas de configuração, tratamento, otimização e proteção, substituindo a execução de tarefas complexas e rotineiras por políticas descritas em alto nível pelos usuários e administradores.

Neste trabalho foram exploradas as políticas orientadas a objetivos da Computação Autônoma. Estas políticas representam uma forma de alto nível para especificação comportamental, definindo objetivos a serem perseguidos e, deixando a aplicação ou o *middleware* determinarem as ações necessárias para que sejam alcançados estes objetivos. Por sua vez, políticas baseadas em funções de utilidade estendem as abordagens orientadas a objetivos, contribuindo para determinação do objetivo mais importante, de maior utilidade, em uma dada situação [11].

Enquanto um instrumento capaz de especificar elementos de contexto, atuando como um padrão para descrições, características, configurações e perfis, que poderão ser compartilhados pelas aplicações e serviços na Computação em Grade destacam-se as ontologias. As ontologias podem ser definidas como uma especificação formal e explícita de uma conceituação compartilhada [5], tornando-se um elemento central para o processamento semântico dos dados que caracterizam o estado da região da grade computacional em questão [16].

A decisão pelo emprego de ontologias na concepção do mecanismo proposto se deve ao fato das mesmas terem uma maior expressividade na definição de contexto e adaptações, por possuírem a possibilidade de realizar inferências a partir das informações semânticas, pela facilidade de reutilização

e extensibilidade por novas aplicações ou novas situações de contexto. Além disso, a revisão bibliográfica realizada, registra um crescimento na utilização de ontologias no desenvolvimento de aplicações, especialmente como artefatos de software, com o objetivo de modelar as informações referentes a essas aplicações [17].

O tratamento dinâmico da adaptação à medida que as aplicações são executadas ainda é uma frente de pesquisa ativa na Computação em Grade [9, 4]. A partir destas considerações, a pesquisa apresentada neste artigo contempla a exploração do uso de processamento semântico baseado em Ontologias, bem como Sistemas Autônomos para Controle da Adaptação ao Contexto na Computação em Grade, contribuindo para que seja facilitado o desenvolvimento de aplicações *context-aware*, isto é, aplicações que possuem a capacidade de adaptar-se ao estado do contexto [12].

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 [18].

O objetivo central do trabalho é avaliar o emprego das tecnologias de processamento semântico e Sistemas Autônomos como mecanismos de controle da adaptação ao contexto na Computação em Grade, considerando as principais características e desafios de pesquisa da mesma. A premissa considerada é que a integração destas tecnologias poderá ser uma alternativa utilizada para o desenvolvimento de aplicações distribuídas, além de poder proporcionar facilidades para a concepção, manutenção e reutilização das mesmas, e como consequência promover um atendimento mais ágil às demandas dos usuários. Nesta perspectiva, 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 adaptações, o EXEHDA-DA suporta que sejam definidos requisitos por aplicação. É considerado que vários elementos de contexto podem ser relevantes para uma mesma aplicação, e que a possibilidade de várias adaptações para um mesmo componente de software poderá contribuir para manter e até mesmo melhorar a qualidade da execução quando de eventuais mudanças de contexto.

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. A OntUbi também aglutina as seguintes:
  - **OntContext** - Ontologia da Situação do contexto: representa os contextos coletados, contextos notificados e os contextos de interesse das aplicações.
  - **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.

No modelo ontológico proposto, o contexto é representado pela ontologia denominada OntUbi, onde serão descritos todos os elementos do contexto de interesse para as aplicações. A OWL,*Web Ontology Language*, foi selecionada como a linguagem para definir e instanciar ontologias. Esta escolha considerou os aspectos de maior nível de expressividade, fornecidos pela OWL, seu suporte a metadados RDF (*Resource Description Framework*) e, também, por suas características de abstrações de classes, generalização, agregação, relações de transitividade, simetria e detecção de inconsistências.
Na ontologia OntUbi, os elementos de contexto ou entidades, chamadas de Classes no OWL, possuem três categorias básicas de informações:

- recursos de hardware: dispositivos, periféricos e informações dos nodos e sensores, tais como: localização, memória, capacidade bateria, largura banda, carga da CPU, entre outros;
- recursos de software: descrição do sistema operacional, do *middleware* e aplicações, da política de adaptação da aplicação;
- recursos de usuário: reflete o perfil e, as preferências do usuário.

De modo mais específico, a Política de Adaptação da Aplicação caracterizada na OntAdapt, tem a finalidade de registrar os perfis dos componentes de software das aplicações para a adaptação. Esta ontologia é composta de especificações de regras que regem o comportamento adaptativo para os componentes da aplicação.

Na OntAdapt podem ser definidas políticas para várias aplicações. Cada aplicação pode ser 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, as quais irão constituir as instâncias do relacionamento `ParamTipo_ParamValor`.

A OntAdapt 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 de software das aplicações. O modelo ontológico previsto permite uma evolução incremental das instâncias da OntAdapt, tais como regras, parâmetros, adaptadores e operações [17].
A Figura 1 mostra as classes, os atributos e os relacionamentos entre as classes da Ontologia OntAdapt. Esta ontologia é instanciada e mantida pelo desenvolvedor, através de um *framework*, denominado FWADAPT, o qual tem como objetivo principal a instanciação da Política da Aplicação para adaptação.

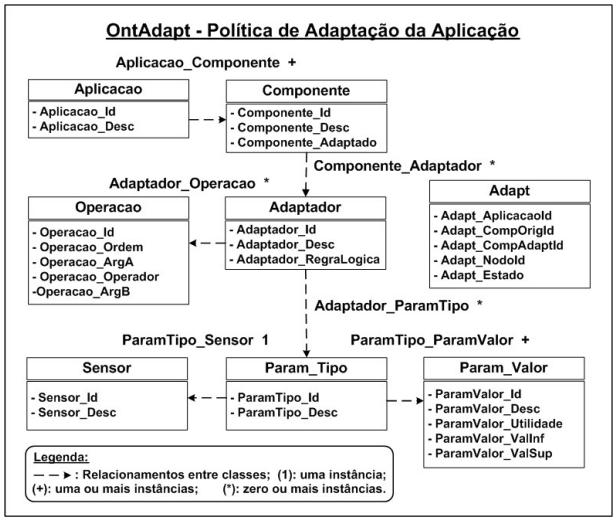

Figura 1. Ontologia da Adaptação

3. EXEHDA-DA: Visão Geral e Modelagem

A concepção do EXEHDA-DA considerou os seguintes aspectos arquiteturais: (i) a monitoração do contexto, a qual deverá ser mantida separada da aplicação e realizada através de elementos reutilizáveis pelo *middleware*; (ii) o processamento de contexto realizado pelo *middleware*, o qual deverá ser extensível e suportar a inclusão de novos elementos de contexto e novas formas de raciocínio. Além da extensibilidade dos elementos do contexto, a separação da execução da aplicação, do controle das suas adaptações, entende-se que deverá contribuir para a reusabilidade de aplicações adaptativas desenvolvidas.

O modelo proposto de controle da adaptação dinâmica de aplicações em ambiente ubíquo, quando da tomada de decisão de adaptação, considera três categorias de informações: (i) o contexto, com base em informações monitoradas, informações semânticas e inferências a partir destas mesmas informações; (ii) a política de adaptação da aplicação; (iii) e as preferências do usuário.

Quanto aos tipos de adaptação dinâmica suportadas pelo EXEHDA-DA, temos [17]:

- **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.

Por sua vez, quanto a estratégia, o EXEHDA-DA contempla uma adaptação Multi-nível Colaborativa organizada em dois níveis [17]. 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 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.

A definição da arquitetura de software do Serviço de Adaptação Dinâmica ao Contexto, EXEHDA-DA (vide figura 2), foi organizada nos seguintes módulos:

- **Tratamento do Contexto Notificado**: recebe do Serviço de Reconhecimento de Contexto, a identificação do contexto notificado. É acessada a definição do contexto de interesse da aplicação, e os valores dos sensores associados ao mesmo. Este mecanismo também consulta a política da aplicação, através do processamento semântico, e acessa as informações necessárias para o cálculo da regra de adaptação em questão. 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 [17], são inferidas as possíveis opções para ação adaptativa, utilizando a regra de adaptação e os valores de parâmetros necessários e oferecidos para calculá-la. As adaptações funcionais e as não-funcionais são computadas da mesma maneira, através da regra de adaptação.

- **Processamento da Regra do Usuário**: acessa 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, retornando ao mesmo um resultado mais significativo para as suas necessidades.

- **Disponibilização da Adaptação Inferida**: neste módulo são retidas as informações necessárias para a ação de adaptação, resultante do 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 do middleware irá acessar as informações para execução da adaptação inferida.

- **Processamento Semântico**: neste módulo foi contemplada a utilização da 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.

4. Estudo de Caso

O modelo de controle de adaptação do EXEHDA-DA por ser configurável por regras específicas por aplicação, permite seu emprego em vários cenários de uso, bem como 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 [17]. Neste trabalho discutiremos o segundo estudo de caso, denominado Alocação Dinâmica de Recursos (ADR), no qual é caracterizado o uso do Controle da Adaptação Dinâmica, enquanto suporte para o escalonamento de recursos em grades computacionais, nas quais os recursos são heterogêneos e sujeitos à taxas de ocupação dinâmicas.

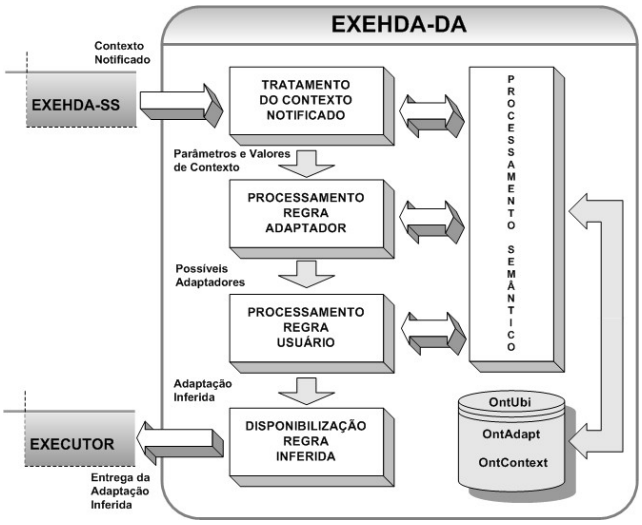

Figura 2. Arquitetura Software EXEHDA-DA

4.1 Objetivos da ADR

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

O ambiente computacional para execução da ADR contempla quatro tipos de software:

- **aplicação paralela CalcPi** - esta aplicação, do tipo *bag-of-tasks*, tem como finalidade matemática calcular o número π (pi) utilizando o método de Monte Carlo. O volume de processamento de cada tarefa a ser disparada é decorrente da precisão desejada para o valor do π;

- **aplicação CpuSteal** - tem como objetivo atuar como um gerador de cargas, baseado em uma distribuição de propriedade exponencial. Este software opera em ciclos de ativação e desativação. Na ativação são geradas operações de ponto flutuante ocupando o processador. Quando inativo, ele entra em repouso. O controle do nível de ocupação média do processador é feito através da determinação de quanto tempo ele permanece ativo e quanto tempo fica em repouso em cada um dos ciclos [14];

- **aplicação LoadGen** - sua finalidade é produzir um descritor de carga computacional. O LoadGen gera os tempos de ativação e desativação de processador, baseado em uma distribuição de probabilidades exponencial, para ser utilizada pelo CpuSteal [14];

- **aplicação MemSteal** - promove a ocupação de memória. Esta aplicação se vale da instanciação de

matrizes para gerar demanda de memória nos nodos processadores.

Os principais objetivos buscados quando da concepção da aplicação ADR podem ser resumidos como:

- ser possível selecionar o recurso computacional mais adequado, segundo um critério de adaptação, entre todos os disponíveis no momento;
- permitir que a distribuição das computações possa ser iniciada, independentemente de ter disponibilidade de processadores para acomodar todas as tarefas paralelas previstas;
- prover tomada de decisão de escalonamento em dois níveis: (i) um contemplando o estado na infraestrutura computacional e (ii) preferências do usuário quanto à origem do recurso;
- modelar a aplicação a ser escalonada de forma a oferecer granulosidade computacional adequada para processamento em grades computacionais;

A grade computacional que constitui a infraestrutura de teste é formada por duas arquiteturas paralelas do tipo *Cluster*. Os nodos destes *clusters* são heterogêneos e estão submetidos a regimes diferenciados de ocupação de memória e ocupação de processador.

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*. Deste modo, o nodo que for inferido como de maior disponibilidade, a partir destes dois critérios, será, então, utilizado para instalação de um componente de software responsável por parte das computações paralelas.

4.2 Adaptações na ADR

O tipo de adaptação a ser tratado pelo EXEHDA-DA para a aplicação Escalonamento de Tarefas é uma adaptação não-funcional (sem mudança de código) com o objetivo de determinar o nodo mais adequado para a execução do componente adaptativo, sendo a escolha do nodo o foco do procedimento adaptativo. A primeira etapa de decisão fornecerá a relação dos melhores nodos, em termos de ocupação de CPU, ocupação de memória e frequência da CPU. A segunda etapa de decisão, definirá a escolha do melhor nodo da relação de acordo com as preferências do usuário. Neste estudo de caso, a preferência do usuário define o *cluster* de interesse do usuário, no caso o *cluster* H3P. Segundo o critério de utilidade do usuário, caso não tenha nodo disponível neste *cluster*, deverão ser escolhidos os melhores nodos disponíveis do *cluster* Langton.

A aplicação CalcPi, utilizando o EXEHDA, instala computações remotas em função da disponibilidade de recursos computacionais na célula de execução. A medida que surjam recursos que atendam os requisitos estabelecidos pelo desenvolvedor na Política de Adaptação (OntAdapt), no âmbito do contexto celular, o componente da aplicação responsável pela gerência da execução é notificado pelo EXEHDA-DA para que execute os procedimentos pertinentes (comando adaptativo `ativa`) que estão após o comando adaptativo `seleciona` no comando adaptativo `noContexto`, conforme a representação na figura 3.

```
int contador = 0;
while (contador < NPP)  {
    sync
    noContexto  200, 000700  {
        ativa     000800, 050;
    }
    contador++;
}


// NPP=Número Processos Paralelos
// 200 = Aplicacao_Id
// 000700 e 000800 = Componente_Id
// 050 = Adaptador_Id
```

Figura 3. Comandos adaptativos

A computação da regra de adaptação deste estudo de caso considerou 5 pressupostos:

- a meta é ativar em 5 nodos (NPP = 5) o algoritmo de cálculo π (CalcPi);
- os 5 nodos serão selecionados de um total de 14 distribuídos entre 2 clusters (H3P e LANGTON);
- foram considerados válidos para o processamento, os nodos com ocupação de CPU inferior a 50%;
- os nodos considerados válidos, serão priorizados de acordo com os valores coletados de frequência de CPU, ocupação de processador e ocupação de memória;
- como resultado da aplicação desta regra de adaptação, foram selecionados 5 nodos.

5. Trabalhos Relacionados

A revisão do estado da arte contemplou o estudo de diversos trabalhos, dos quais 7 foram selecionados em função de sua similaridade para discussão nesta seção. Uma avaliação de todos os trabalhos relacionados está disponível em [17].

A tabela 1 apresenta uma comparação entre os trabalhos relacionados, 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.

MODELOS	01	02	03	04	05	06	07	08
CARISMA	+		+		+	+	+	+
CHISEL		+	+				+	
QUO	+		+			+		
RAINBOW	+		+			+	+	
MADAM	+	+	+		+		+	+
PROTEUS	+		+	+			+	
SECAS	+		+		+		+	
EXEHDA-DA	+	+	+	+	+	+	+	+

A seguir uma breve breve descrição dos trabalhos comparados na tabela 1:

- **Carisma** [1], 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.

- **Chisel** [10], 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.

- **QUO** (Objetos de Qualidade) [8], 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.

- **Rainbow** [6], 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.

- **Madam** [7], 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.

- **Proteus** [15], 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.

- **SECAS** [2], *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.

A função de utilidade considera os parâmetros de adaptação da aplicação e do contexto de execução, bem como as preferências do usuário. A função de utilidade calcula a soma ponderada das diferenças entre os parâmetros ofertados pelo ambiente e os necessários pela aplicação, onde os pesos refletem as prioridades das necessidades do usuário. Dentro deste modelo, o objetivo do Serviço EXEHDA-DA pode ser formulado como a garantia de que, a qualquer momento, será executada a adaptação com a mais alta utilidade e que não viole as limitações de recursos impostos pelo ambiente.

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

O emprego de um modelo ontológico personalizável por aplicação permite que todo contexto mensurável possa 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 arquitetura de software do EXEHDA-DA, baseada em

regras de adaptação personalizáveis por aplicação, agiliza o desenvolvimento de aplicações adaptativas, através da independência da aplicação, tanto em relação ao modelo de controle da adaptação, quanto à especificação da Política de Adaptação.

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.

As políticas de adaptação serão expressas utilizando funções de utilidade, e promovem uma adaptação (que pode ser composta por diversas outras - composicional) direcionada aos componentes de software da aplicação. A adaptação precisa conhecer a estrutura de componentes e limitações da aplicação, bem como os vários contextos e dependências de recursos. Isto implica em que o *middleware* trabalhe, em tempo de execução, em um modelo arquitetônico, onde as aplicações especificam previamente todas as variantes e limitações de adaptação. O EXEHDA-DA realizará, de maneira dinâmica, adaptações funcionais e não-funcionais.

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

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.

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.

Referências

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

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

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

[4] F. J. da Silva e Silva, F. Kon, A. Goldman, M. Finger, R. Y. de Camargo, F. C. Filho, and F. M. Costa. Application execution management on the integrade opportunistic grid middleware. *J. Parallel Distrib. Comput*, 70(5):573–583, 2010.

[5] D. Fensel. Ontologies-silver bullet for knowledge management and electronic commerce, 2000.

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

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

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

[9] K. Henricksen, J. Indulska, and A. Rakotonirainy. *Using context and preferences to implement self-adapting pervasive computing & applications*. 2006.

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

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

[12] F. Kon, J. R. Marques, T. Yamane, R. H. Campbell, and M. D. Mickunas. Design, implementation, and performance of an automatic configuration service for distributed component systems. *Softw, Pract. Exper*, 35(7):667–703, 2005.

[13] J. J. O. Neto and F. M. Costa. An interceptor model to provide dynamic adaptation in the integrade. *Anais do VII Workshop on Grid Computing and Applications*, 2009.

[14] R. A. Real. Tips, uma proposta de escalonamento direcionada à computação pervasiva, 2004. [s.n.].

[15] A. Toninelli, R. Montanari, L. Kagal, and O. Lassila. Proteus: A semantic context-aware adaptive policy model. *IEEE Computer Society*, pages 129–140, 2007.

[16] A. C. T. Vidal, F. J. da Silva e Silva, S. T. Kofuji, and F. Kon. Applying semantics to grid middleware. *Concurrency and Computation: Practice and Experience*, 21(13):1725–1741, 2009.

[17] N. Warken. 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, 2010. Disponível: http://olaria.ucpel.tche.br/nelsiw/.

[18] A. C. Yamin. 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, 2004.