Serviço de adaptação de conteúdo para aplicações multiplataforma

Diego Carvalho[1], Fernando Trinta[1]

[1]Universidade de Fortaleza - UNIFOR
Av. Washington Soares, 1321, Bloco J, Sala 30
CEP 60811-905, Fortaleza-CE, Brasil

diego.carvalho@edu.unifor.br, trinta@unifor.br

Abstract. *The increasing usage of heterogeneous devices in our ordinary activities has forced current applications to adapt their data visualization according to context information, such as the device itself or even user preferences. This paper presents a generic service that performs content adaptation for multiplatform applications. This service aims to be modular and non-instrusive. In other words, the content adaptation logic is considered a crosscutting concern according to the Aspect Oriented Paradigm. Our proposal presents a software development process and the service architecture. Finally, a case study and its results are also presented.*

Resumo. *A expansão do uso de dispositivos heterogêneos para acesso ubíquo requer que aplicações adaptem a visualização de seus dados de acordo com características contextuais, como dispositivo ou preferências pessoais do usuário. Este artigo apresenta uma proposta de um serviço genérico de adaptação de conteúdo para aplicações multiplataforma. Este serviço objetiva ser modular, não intrusivo à lógica de negócios da aplicação, tratando a adaptação de conteúdo como um interesse transversal a qualquer aplicação, de acordo com o paradigma de orientação a aspectos. São apresentados um processo guia de uso do serviço, seus principais componentes e os resultados iniciais de seu uso através de um estudo de caso.*

1. Introdução

Computação Pervasiva ou Ubíqua representa um novo paradigma computacional que busca promover uma computação melhor integrada ao cotidiano de seus usuários. Sistemas de computação pervasiva propõem o acesso constante do usuário às suas aplicações e dados via diferentes dispositivos e diferentes redes de comunicação sem fio, em que dados contextuais são monitorados pelas aplicações ou por sistemas de suporte, que modificam seu comportamento em função de mudanças relevantes nestas informações, para melhor atender às requisições do usuário [12].

Dentre as características mais relevantes dos sistemas de computação pervasiva está o uso de diferentes dispositivos para acessar uma mesma aplicação. Estes dispositivos possuem características heterogêneas, tais como diferentes tamanhos de tela, capacidade de memória/processamento, facilidades de entrada/saída e de conectividade. Este cenário exposto cria a exigência de que as informações de aplicações pervasivas precisem ser adaptadas, levando-se em consideração estes

dispositivos. Na realidade, o dispositivo é apenas umas das variáveis que devem ser consideradas na adaptação das informações por uma aplicação pervasiva. Outras características relevantes incluem preferências do usuário, local ou momento do uso, etc. Este conjunto das informações relevantes a uma aplicação pervasiva é definido como contexto [1], e aplicações influenciadas em seu comportamento por informações contextuais são classificadas como aplicações cientes de contexto (*context-aware applications*).

Em geral adaptação de conteúdo envolve transformações de dados multimídia como a adaptação de formatos e tipos, tradução, compressão dos dados, agregação/desagregação dos dados, dentre outras [5]. Porém, não está limitada a estas possibilidades, podendo ser definidas adaptações específicas a determinado sistema ou dispositivo.

Boa parte das soluções proposta de adaptação de conteúdo utiliza o paradigma *middleware* e são voltadas para desenvolvimento de aplicações pervasivas para domínios específicos como multimídia, jogos ou medicina [12] [6]. Logo, não há uma preocupação em generalizá-las. Mais que isso, em sua maioria, os serviços utilizados obrigam o uso de um modelo de programação que mistura as funcionalidades básicas das aplicações com uma API fornecida por eles de forma intrusiva. Outros trabalhos apresentam soluções de *middleware* para a construção de aplicações pervasivas cientes de contexto, de forma genérica [11] [4]. Tais soluções procuram contornar o problema da intrusão do código dos serviços do *middleware* nas funcionalidades da aplicação utilizando o paradigma de programação orientada a aspectos [9], o qual propõe o encapsulamento de comportamentos transversais em estruturas de programação denominadas aspectos, que podem ser combinadas com os componentes que implementam a lógica de negócio da aplicação através de mecanismos de composição. Porém, as soluções de *middleware* citadas, não especificam o uso da tecnologia de aspectos diretamente na adaptação de conteúdo.

Este trabalho propõe soluções para os dois problemas apresentados previamente, a saber, *(i)* o fato de serviços de adaptação de conteúdo serem voltados a domínios específicos, e *(ii)* o modelo de programação intrusivo destes serviços. Para isso, apresenta *(i)* um processo de adaptação de conteúdo para aplicações pervasivas e *(ii)* um serviço genérico que executa este processo de adaptação e pode ser acoplado a tais aplicações de forma não intrusiva, ou seja, sem exigir refatorações diretas do código relativo a lógica de negócios de cada aplicação. Almeja-se com isso melhorar a flexibilidade de aplicações pervasivas, como também permitir que aplicações legadas possam utilizar o serviço para se tornarem multiplataforma, com o menor esforço possível de alterações. Para o desenvolvimento do serviço foi usado o paradigma de programação orientada a aspectos.

O corpo deste artigo é dividido em seções. Após esta introdução, na seção 2, é feito um apanhado geral de trabalhos relacionados com o tema proposto. Em seguida, na seção 3, são apresentados o processo e o serviço de adaptação de conteúdo propostos. Logo após, na seção 4, é apresentado o estudo de caso realizado para analisar as etapas do processo e o serviço implementado, e discutidos os resultados obtidos. Por fim, na seção 5, é feita a conclusão e são apresentados os trabalhos futuros.

2. Trabalhos Relacionados

No levantamento de estado da arte, buscou-se informação a respeito de trabalhos relacionados a plataformas de *middleware* que ofereçam serviços voltados à adaptação de conteúdo, visando analisar suas principais limitações e características relevantes.

Vários trabalhos definem plataformas de *middleware* para construção de aplicações pervasivas voltadas a domínios de negócio específicos, como medicina, jogos, e outros. Tais plataformas adotam uma arquitetura baseada em serviços, dentre os quais está a adaptação de conteúdo.

Um dos trabalhos estudados propõe uma camada de serviços especializados na forma de um *middleware* para oferecer suporte a jogos multiusuário multiplataforma. Entre estes serviços está a adaptação de conteúdo, que objetiva transformar as informações relevantes do jogo para um formato adequado ao contexto de execução atual do jogador [12]. Outro trabalho, chamado *Ubidoctor*, também propõe uma camada de serviços para a construção de aplicações pervasivas voltadas à área médica. Dentre os serviços oferecidos consta a adaptação de conteúdo, cujo objetivo é prover transformações para as informações migradas entre sessões de uso do ambiente de medicina ubíquo [6].

Analisando os trabalhos acima mencionados, identifica-se como uma limitação das propostas, a impossibilidade de generalização destas para utilização em diferentes domínios de aplicação, uma vez que lidam com elementos específicos do domínio de aplicação tratado. Outra limitação percebida é o atrelamento do código responsável pela execução da adaptação ao código de negócio da aplicação desenvolvida usando-se o *middleware*.

A análise destes trabalhos também permitiu destacar pontos relevantes, aproveitados na proposta deste artigo. Ambos usam a idéia de um procedimento de adaptação de conteúdo, composto por passos sequenciais que devem ser executados para que a adaptação seja realizada de maneira uniforme e o conceito de estratégia ou política de adaptação, que serve para determinar quais transformações (procedimentos lógicos) devem ser realizadas nos dados e a melhor maneira de realizá-las.

Em relação a *context-awareness*, foi identificado que alguns trabalhos já abordam soluções de *middleware* para construção de aplicações adaptativas cientes do contexto que podem ser aplicadas a diferentes domínios de negócio. O PACCA (*Projeto de Aplicações Ciente ao Contexto e Adaptativas*) propõe um arcabouço para desenvolvimento e execução de aplicações adaptativas cientes do contexto, no qual a programação orientada a aspectos é usada para modularizar o comportamento adaptativo e dissociá-lo da lógica de adaptação [11]. Ele define um módulo de adaptação, que é o componente responsável por gerenciar todo o processo de adaptação dinâmica, na forma de um aspecto abstrato, visando um maior desacoplamento e flexibilidade da arquitetura proposta, uma vez que esta pode ser estendida com a inclusão de novos aspectos concretos para representar comportamentos adaptativos adicionais. Estes aspectos concretos definem vários interesses adaptativos, tais como adaptação das telas de acordo com os dispositivos utilizados, adaptação a diferentes tipos e condições de rede, entre outros.

Destaca-se no PACCA o uso dos aspectos como elemento base da implementação do módulo de adaptação, o que foi também aplicado neste trabalho para o desenvolvimento do serviço de adaptação de conteúdo genérico. Todavia, a diferença fundamental entre o PACCA e este trabalho é que no PACCA não é destacado o interesse de adaptação do conteúdo a ser apresentado ao usuário, que é o foco do serviço construído neste trabalho.

3. Proposta de Solução

A proposta apresentada neste artigo busca tratar as limitações identificadas nos trabalhos relacionados ao tema, que são *(i)* o fato de serviços de adaptação de conteúdo serem voltados a domínios específicos e *(ii)* o modelo de programação intrusivo destes serviços. Para tanto, esta proposta é baseada em dois pilares: *(i)* a definição de um processo de adaptação de conteúdo para aplicações multiplataforma e *(ii)* a implementação de um serviço de adaptação de conteúdo genérico, que execute conforme o processo definido, e possa ser integrado às aplicações de forma simples e não intrusiva, ou seja, sem necessidade de alterações diretas no código da lógica de negócios do sistema. O objetivo é melhorar a flexibilidade de aplicações pervasivas, e permitir que aplicações legadas possam utilizar o serviço genérico para se tornarem multiplataforma, com o menor esforço possível no que diz respeito a alterações.

3.1. Processo de Adaptação de Conteúdo

Tendo como base a idéia apresentada nos trabalhos relacionados [12] [6], que definem os passos necessários para realização da adaptação de conteúdo nas plataformas de *middleware* que propõem, este trabalho definiu um processo de adaptação de conteúdo genérico, aplicável a qualquer sistema que deseje se tornar multiplataforma. Foram definidas, conforme descrito a seguir, seis etapas para o processo. A definição destes passos serviu como levantamento dos requisitos a serem contemplados pelo serviço genérico de adaptação de conteúdo.

Etapa 1. Definir procedimentos lógicos de adaptação.

O processo de adaptação de conteúdo tem como base transformações executadas sobre os dados, tais como a adaptação de formatos e tipos, tradução, compressão dos dados, agregação/desagregação dos dados, dentre outras. Podem ser definidas também adaptações específicas a determinado sistema ou dispositivo [10]. Estas transformações, chamadas neste trabalho de procedimentos lógicos de adaptação, podem ser oferecidas pelo serviço já implementadas. Exemplos de possíveis procedimentos lógicos são os relacionados na tabela 1.

Tabela 1. Exemplos de Procedimentos Lógicos de Adaptação.

TIPO DE DADO	PROCEDIMENTOS LÓGICOS
Texto	Sumarização (retirada de palavras); Tradução.
Áudio	Conversão entre formatos.
Vídeo	Conversão entre formatos; Transformação em Imagens.
Imagem	Redução/Aumento do tamanho; Modificação de cores.

O desenvolvedor deve poder, caso necessite, executar esta etapa, na qual define novos procedimentos lógicos específicos para sua aplicação. Desta forma, o serviço deve oferecer um mecanismo de definição destes procedimentos que seja extensível.

Etapa 2. Definir plataformas-alvo para as quais o conteúdo da aplicação deve ser adaptado.

Um elemento essencial para o processo de adaptação de conteúdo são as possíveis plataformas de hardware e conexão que serão usadas para executar a aplicação usuária do serviço. Esta etapa objetiva definir a granularidade das informações que se deseja tratar sobre estas plataformas. Estas definições servirão como base para o desenvolvedor estender o mecanismo de definição de plataformas-alvo, apresentado pelo serviço de adaptação de conteúdo, da forma mais adequada para a aplicação que construirá.

Etapa 3. Definir estratégias de adaptação.

Outro conceito associado ao funcionamento do processo de adaptação de conteúdo é o de estratégia de adaptação, retirado dos trabalhos analisados [12] [6], e que diz respeito à definição de quais transformações devem ser realizadas nos dados e a melhor maneira de realizá-las.

Nesta etapa, devem ser definidas as estratégias de adaptação a serem executadas pelo serviço sobre os conteúdos adaptáveis. Na definição, selecionam-se quais procedimentos lógicos devem ser executados e a quais plataformas-alvo associá-la. O serviço deve apresentar mecanismo de construção de estratégias de adaptação.

Etapa 4. Configurar os relacionamentos entre os procedimentos lógicos, estratégias de adaptação e plataformas-alvo.

Esta etapa objetiva associar os elementos definidos nas etapas 1, 2 e 3 para garantir a atuação correta do serviço sobre a aplicação. A partir dos relacionamentos definidos nesta etapa, o serviço de adaptação de conteúdo consegue executar sobre as requisições feitas à aplicação e executar o procedimento adequado sobre os dados adaptáveis constantes nas respostas da aplicação a estas requisições. Logo, o serviço deve oferecer mecanismo de configuração dos relacionamentos definidos.

Etapa 5. Definir mapeamento do conteúdo original para o conteúdo como visto na API do serviço.

Esta etapa é necessária pois o serviço de adaptação de conteúdo define e trabalha com tipos de dados genéricos, tais como, texto, gráfico (imagem), multimídia (Áudio/Vídeo), binário, e outros. Desta forma, os conteúdos mapeados na etapa 6 precisam ser convertidos de/para estes tipos de dados genéricos, os quais são manipulados pelas estratégias de adaptação e procedimentos lógicos. A API do serviço deve oferecer, portanto, um mecanismo para que o usuário implemente as conversões de/para os tipos genéricos definidos.

Etapa 6. Indicar quais informações da aplicação compõem o conteúdo a ser adaptado para diferentes plataformas.

Esta etapa consiste em identificar entre os elementos de projeto da aplicação aqueles que representam os conteúdos manipulados em seu ciclo de execução e

precisam sofrer adaptações de acordo com o contexto de execução. O serviço deve então oferecer um mecanismo não intrusivo para indicação deste mapeamento.

3.2. Arquitetura do Serviço de Adaptação de Conteúdo

O serviço de adaptação de conteúdo pretende contornar as limitações apresentadas no início da seção 3 e é aderente às etapas do processo apresentado na seção 3.1. Basicamente, ele é composto por uma API de classes e aspectos abstratos, que permitem a realização das etapas do processo de adaptação de conteúdo para uma aplicação, e por um aspecto principal, que controla a execução do serviço sobre esta aplicação. O serviço foi implementado com o uso das linguagens Java e Aspectj [8]. Na figura 1 é apresentado o diagrama de classes do serviço de adaptação de conteúdo.

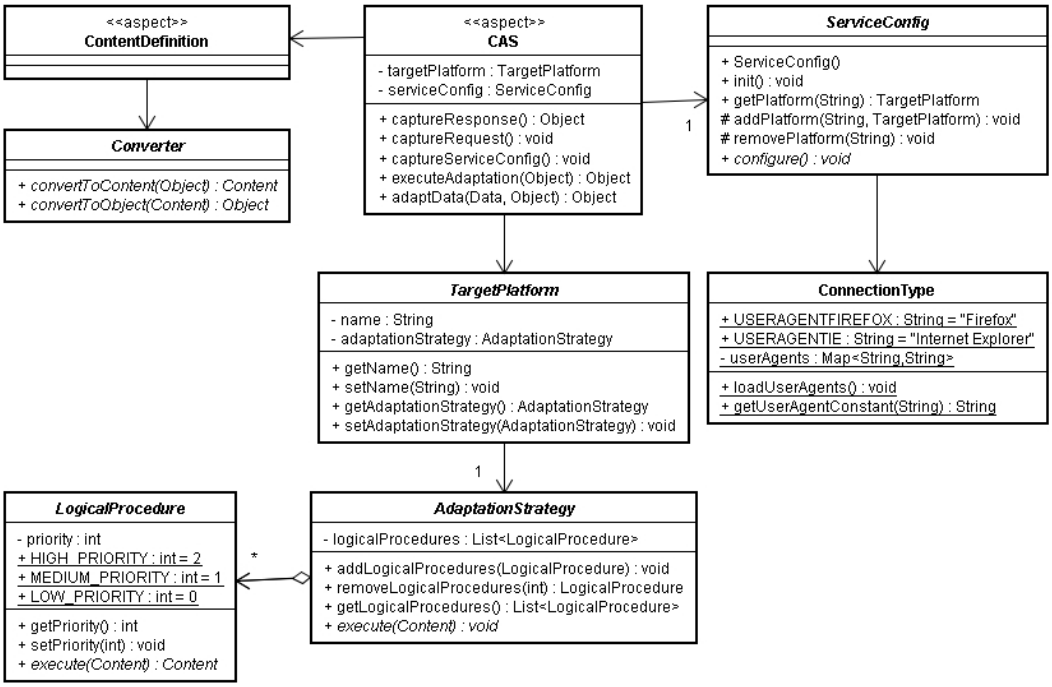

Figura 1. Diagrama de classes do serviço de adaptação de conteúdo.

O uso de programação orientada a aspectos na implementação do serviço de adaptação de conteúdo visa garantir a separação deste interesse transversal da lógica de negócios da aplicação usuária, bem como permitir a integração do serviço de forma simples em aplicações legadas.

Na programação orientada a aspectos existe um conceito fundamental chamado *join points*, que são elementos semânticos da linguagem com os quais os aspectos se coordenam [9], ou seja, são pontos do fluxo de execução de um programa capturados pelos aspectos. Na linguagem AspectJ [8] são associados ao conceito de *join points* os conceitos de *pointcuts* e *advices*. Os *pointcuts* servem para capturar *join points* no fluxo de execução dos programas, e os *advices* para definir o comportamento transversal a ser executado quando um *pointcut* é atingido.

A explicação da arquitetura é dividida da seguinte forma: na seção 3.2.1 é detalhada a API do serviço que deve ser usada pelo desenvolvedor para executar as

etapas do processo de adaptação de conteúdo; na seção 3.2.2 é apresentado o aspecto CAS, que controla a execução do serviço sobre a aplicação.

3.2.1. API do Serviço de Adaptação de Conteúdo

O uso do processo de adaptação de conteúdo por uma aplicação ocorre através da implementação da API oferecida pelo serviço de adaptação de conteúdo. Estas classes representam os conceitos associados às etapas do processo.

Inicialmente, é definido um conjunto de tipos de dados genéricos na API do serviço, os quais representam a forma como os conteúdos são manipulados pelos procedimentos lógicos e estratégias de adaptação executados por ele. A classe base da hierarquia é a classe abstrata *Content,* que representa um conteúdo genérico que pode ser adaptado, como texto, imagem, etc. Os tipos já oferecidos pelo serviço herdam desta classe, conforme figura 2. Caso seja necessário definir um novo tipo genérico, deve-se acrescentá-lo a esta hierarquia de herança.

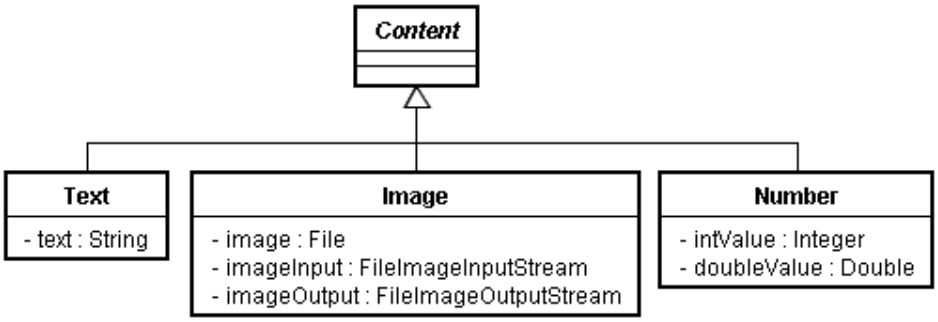

Figura 2. Tipos de dados genéricos do serviço de adaptação de conteúdo.

Os procedimentos lógicos de adaptação são representados na API do serviço pela classe abstrata *LogicalProcedure*, exibida na figura 1, que deve ser estendida e ter o método *execute* implementado. Este método recebe como parâmetro um objeto do tipo *Content*, executa determinado procedimento adaptativo sobre este, e retorna o objeto *Content* adaptado. Nesta classe é definido ainda um atributo chamado *priority*, que permite definir prioridades diferentes para a ordem de execução dos diversos procedimentos lógicos implementados.

Uma estratégia de adaptação é representada pela classe abstrata *AdaptationStrategy*, conforme figura 1, que deve ser estendida e ter o método *execute* implementado. Esta classe possui como atributo uma lista de objetos *LogicalProcedure*. No método *execute*, devem ser definidos quais procedimentos lógicos aplicar, e a melhor forma de fazê-lo, sobre o parâmetro recebido, que é um objeto *Content*.

Uma plataforma-alvo é representada pela classe abstrata *TargetPlatform*, exibida na figura 1, que deve ser estendida para cada plataforma que possa acessar a aplicação. Uma *TargetPlatform* possui como atributo essencial um objeto *AdaptationStrategy*, ou seja, cada plataforma-alvo deve ter associada a si uma estratégia de adaptação de conteúdo.

A API do serviço oferece a classe abstrata *ServiceConfig*, que permite configurar os relacionamentos entre os procedimentos lógicos, estratégias de adaptação e

plataformas-alvo. Esta classe deve ser estendida e ter o método *configure* implementado. Nele devem ser instanciados objetos *LogicalProcedure*, *AdaptationStrategy* e *TargetPlatform*, bem como devem ser definidas as associações entre eles. As *TargetPlatform* instanciadas devem ser adicionadas ao atributo *platforms*, um objeto mapa que associa uma *TargetPlatform* a uma chave que representa uma plataforma-alvo usada para acesso à aplicação.

Algumas chaves são pré-definidas na classe utilitária *ConnectionType* e outras podem ser adicionadas. Estas chaves devem ser associadas aos valores do parâmetro "*user-agent*" de uma requisição HTTP à aplicação. Os pares *user-agent* e chave associada devem ser adicionados no atributo mapa *userAgents* de *ConnectionType*. O serviço captura as requisições e, a partir do valor deste parâmetro, busca a chave associada à plataforma-alvo que está sendo usada no acesso à aplicação. A partir da chave, pode recuperar da subclasse que estendeu *ServiceConfig*, a *TargetPlatform* a ser utilizada no processo de adaptação de conteúdo.

Um detalhe importante a ressaltar é que, na versão atual do serviço, a classe *ServiceConfig* estende a classe *HttpServlet* da linguagem Java. Logo, a classe que estende-la, deve ser mapeada como *servlet* da aplicação, uma vez que o serviço captura-a quando esta é carregada pelo servidor e inicia a configuração definida nela.

A classe abstrata *Converter*, da figura 1, possui os métodos abstratos *convertToContent* e *convertToObject*, que devem ser implementados para cada classe que a estenda, pois definem, respectivamente, a conversão para um tipo genérico da API do serviço, e de um tipo genérico da API do serviço para um objeto de determinada classe de conteúdo da aplicação. Estas classes serão associadas aos dados de negócio da aplicação que forem mapeados como adaptáveis.

Por fim, a API do serviço oferece um aspecto abstrato, chamado *ContentDefinition*, e propõe o uso dos chamados *annotations types* [7], que são usados para expressar metadados sobre os membros de uma aplicação (classes, atributos, métodos, etc.) na forma de anotações inseridas no código fonte. Atualmente, são propostas para o serviço as anotações da tabela 2.

Tabela 2. Anotações do Serviço de Adaptação de Conteúdo.

Anotação	**Definição**
@BusinessClass	Marca uma classe de negócio da aplicação usuária do serviço.
@Content	Marca uma classe de conteúdo da aplicação usuária do serviço.
@Data (type, converter)	Define os atributos de uma classe marcada com @Content que representam dados que devem sofrer adaptação. O parâmetro *type* indica o tipo genérico do serviço que é associado ao conteúdo do atributo, e o parâmetro *converter* indica qual *Converter* deve ser usado para realizar o mapeamento tipo específico / tipo genérico e vice-versa.

Através deste aspecto, o desenvolvedor pode mapear os tipos de dados específicos de uma aplicação para os tipos genéricos utilizados pelo serviço, através de anotações. Um exemplo deste mapeamento será visto a seguir na exposição do estudo de caso (seção 4.1).

3.2.2. Execução do Serviço de Adaptação de Conteúdo

A execução do serviço é controlada pelo aspecto principal, denominado CAS, um aspecto concreto que define alguns *pointcuts* e seus respectivos *advices* associados, bem como métodos responsáveis pela execução do processo de adaptação de conteúdo.

O primeiro *pointcut* definido chama-se *captureServiceConfig* e é responsável por capturar a inicialização estática da classe da aplicação que estende e implementa a classe de configuração do serviço (*ServiceConfig*). Sua execução é ilustrada na Figura 3. Esta classe é mapeada como um *servlet* da aplicação. Ao capturar o *poincut*, instancia-se a classe capturada e atribui-se o objeto ao atributo *serviceConfig* do aspecto CAS. Em seguida, é executado o método *init* de *serviceConfig*, para garantir a inicialização das configurações do serviço para a aplicação. O fluxo de execução se completa carregando-se a lista de valores "*user-agent*" cadastrada na classe utilitária *ConnectionType*, através da chamada ao método *loadUserAgents* desta classe.

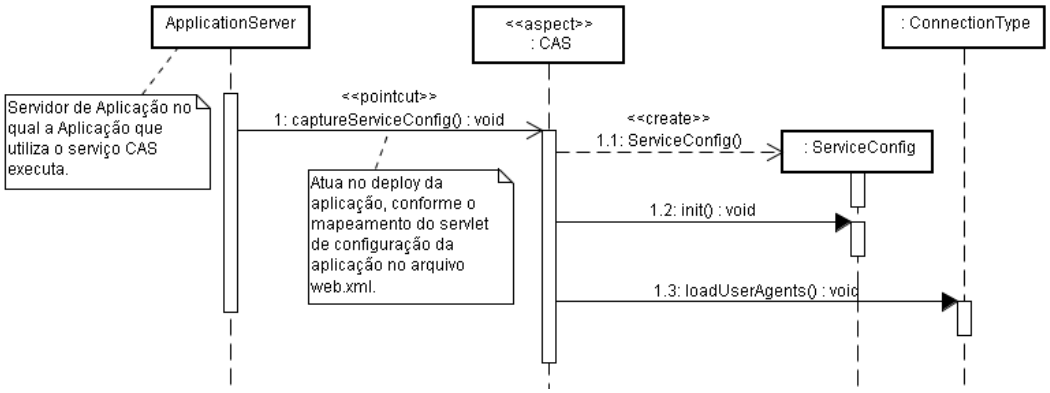

Figura 3. Atuação sobre o pointcut *captureServiceConfig* do aspecto CAS.

Outro *pointcut* definido é o *captureRequest*, cujo o fluxo de execução é representado pelo diagrama de seqüência da Figura 4. Este *pointcut* tem a função de capturar as requisições HTTP feitas à aplicação. A partir da requisição capturada, um *advice* executa e recupera o valor do parâmetro "*user-agent*", buscando, a partir dele, em *serviceConfig* a *TargetPlatform* associada à plataforma-alvo usada na requisição. A partir do retorno recebido, o atributo *targetPlatform* do aspecto CAS é configurado.

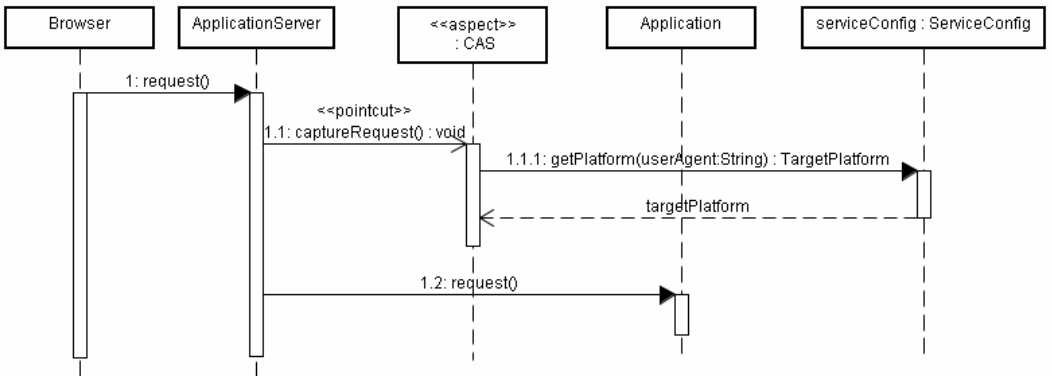

Figura 4. Atuação sobre o pointcut *captureRequest* do aspecto CAS.

O último *pointcut*, chamado *captureResponse*, é aquele que dispara a execução da adaptação de conteúdo em si, pois captura as execuções dos métodos das classes de negócio (aquelas marcadas com a anotação @*BusinessClass*) que retornam instâncias das classes de conteúdo (aquelas marcadas com a anotação @*Content*). Quando o *pointcut* é atingido, um *advice* captura o objeto da classe de conteúdo que está sendo retornado e chama a execução do método *executeAdaptation* passando-o como parâmetro. Após o retorno do método, que é o objeto adaptado, o *advice* libera a finalização do envio da resposta do método de negócio da aplicação. A execução sobre este *pointcut* é apresentada na figura 5.

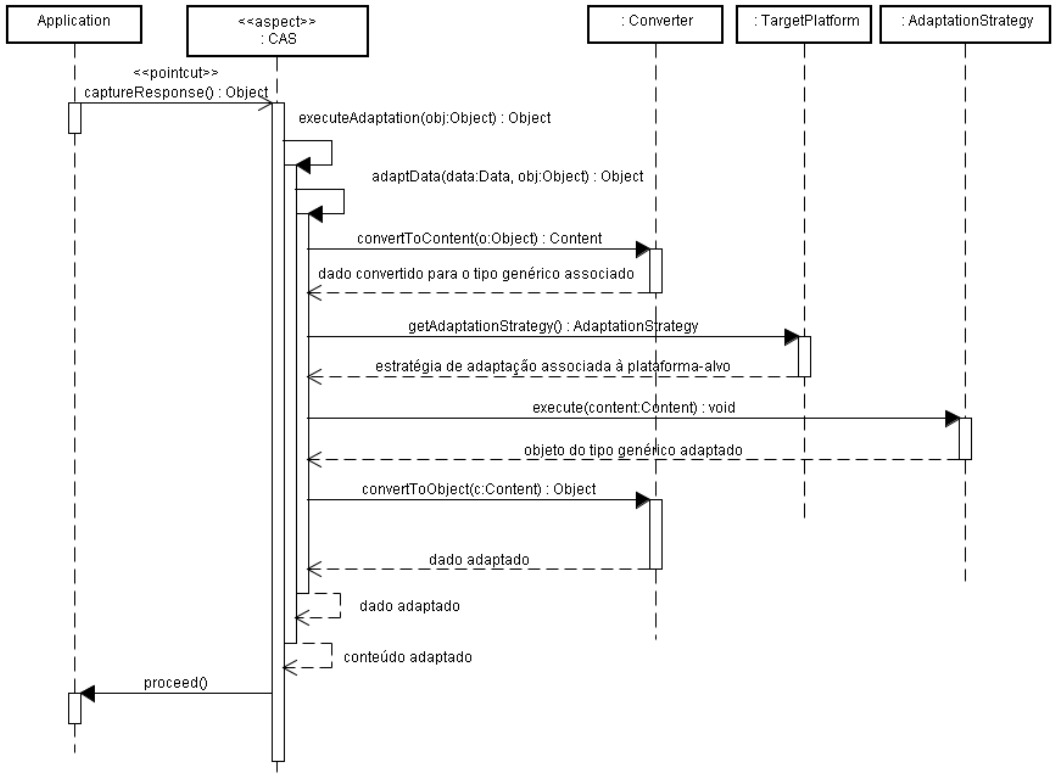

Figura 5. Atuação sobre o pointcut *captureResponse* do aspecto CAS.

O método *executeAdaptation* é responsável por recuperar, a partir do objeto instância de uma classe de conteúdo da aplicação (marcada com @*Content*) recebido como parâmetro, os atributos marcados como dados adaptáveis (marcados com @*Data*). Para cada um destes atributos é chamada a execução do método *adaptData* e, de acordo com o retorno recebido, configurado o dado com o valor adaptado.

O método *adaptData*, que recebe como parâmetro o dado e a anotação @*Data* associada a ele, chama inicialmente o método *convertToContent* do *Converter* associado. O objeto convertido para o tipo genérico do serviço é então passado para o método *execute* da *AdaptationStrategy* associada ao atributo *targetPlatform* do aspecto CAS. Em outras palavras, o objeto que representa conteúdo sofre as adaptações adequadas de acordo com a estratégia de adaptação associada à plataforma utilizada pelo usuário. O objeto retornado é enviado então como parâmetro para o método *convertToObject* do *Converter*, e o objeto adaptado é retornado.

4. Estudo de Caso

Para verificar o processo proposto e a implementação do serviço de adaptação de conteúdo, aplicou-se um estudo de caso desenvolvendo uma aplicação web em Java. A aplicação desenvolvida simula conceitos da área médica e utiliza a arquitetura MVC (*Model-View-Controller*) [3].

Inicialmente, foram definidos os papéis das classes da aplicação, exibidas na figura 6, em relação ao serviço de adaptação de conteúdo. A classe *Prontuario*, que modela um prontuário médico de um paciente, foi selecionada como classe de conteúdo e seus atributos foram selecionados como dados adaptáveis. A classe *ProntuarioBC*, que trabalha como *BusinessController*, foi selecionada como classe de negócio da aplicação, cujos métodos retornam objetos das classes de conteúdo que devem ser adaptados pelo serviço. Por fim, a classe *servlet ProntuarioTest* e algumas páginas JSP serviram para simular a execução de requisições à aplicação.

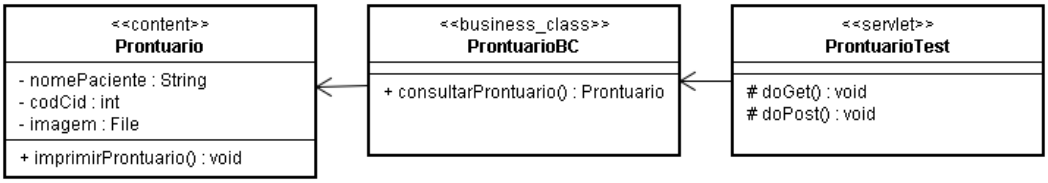

Figura 6. Classes da aplicação experimental – Prontuário Médico.

Os objetivos principais do estudo de caso foram: *(i)* verificar a aplicabilidade do processo na definição da adaptação de conteúdo para aplicações pervasivas usuárias do serviço de adaptação de conteúdo; *(ii)* verificar o atendimento da API oferecida pelo serviço às etapas do processo; *(iii)* Verificar a execução do serviço sobre a aplicação.

4.1. Aplicabilidade do processo de adaptação de conteúdo e atendimento da API do serviço para implementação das etapas deste processo.

Inicialmente foram definidos e implementados alguns procedimentos lógicos para a aplicação, tais como concatenação de textos (*ConcatNameLogicalProcedure*) e redução de cores em imagens (*ImageLogicalProcedure*) . Em seguida, com base na definição das plataformas que enviariam requisições à aplicação, foram definidas as estratégias de adaptação e plataformas-alvo. Para computadores *desktop* foram implementadas as classes *DesktopAdaptationStrategy* e *DesktopTargetPlatform*. Para dispositivos móveis foram implementadas as classes *MobileDeviceAdaptationStrategy* e *MobileDeviceTargetPlatform*.

Na seqüência da execução do processo, foi implementada a classe *AppServiceConfig*, de configuração da aplicação para o serviço. Nela foram instanciados os *LogicalProcedures*, *AdaptationStrategies* e *TargetPlatforms* implementados anteriormente, e seus relacionamentos definidos. Para computadores *desktop* não foram necessárias adaptações no conteúdo, logo não foi associado nenhum *LogicalProcedure* à estratégia de adaptação associada, *DesktopAdaptationStrategy*. Já para dispositivos móveis foram necessárias adaptações no conteúdo, logo os procedimentos lógicos implementados, como concatenação de textos e redução de cores em imagens, foram associados a *MobileDeviceAdaptationStrategy*.

Foram implementados ainda para a aplicação, os *Converters* necessários para transformar os tipos de objetos específicos da aplicação para os tipos de dados genéricos da API do serviço e vice-versa. Encerrando a execução do processo sobre a aplicação, foi implementado o aspecto *AppContentDefinition*, que definiu a marcação das classes da aplicação com as anotações do serviço. Esta marcação teve como base os papéis definidos para as classes da aplicação, conforme explicado no início da seção 4. Por exemplo, a classe *Prontuario* foi marcada com a anotação *@Content* para indicar que representa uma classe de conteúdo da aplicação, como pode ser visto na figura 7, que apresenta o código do aspecto.

```
public aspect AppContentDefinition extends ContentDefinition{
  declare @type: Prontuario : @br.unifor.mia.cas.annotations.Content;
  declare @type: ProntuarioBC : @br.unifor.mia.cas.annotations.BusinessClass;
  declare @field: * Prontuario.nomePaciente :
@br.unifor.mia.cas.annotations.Data(type="br.unifor.mia.cas.classes.content.Text",
converter="br.unifor.mia.app.converter.TextConverter");
  declare @field: * Prontuario.codCid :
@br.unifor.mia.cas.annotations.Data(type="br.unifor.mia.cas.classes.content.Number",
converter="br.unifor.mia.app.converter.NumberConverter");
  declare @field: * Prontuario.imagem :
@br.unifor.mia.cas.annotations.Data(type="br.unifor.mia.cas.classes.content.Image",
converter="br.unifor.mia.app.converter.ImageConverter");
}
```

Figura 7. Aspecto de mapeamento do conteúdo da aplicação AppContentDefinition.

4.2. Atuação do serviço de adaptação de conteúdo sobre a aplicação.

A execução da aplicação ocorreu através do envio de requisições à aplicação, contento os mesmos dados, a partir do formulário na página inicial. Os dados da requisição eram tratados pela classe *ProntuarioTest*, que acionava o método *consultarProntuario* da classe de negócio *ProntuarioBC*. Este método retornava um objeto da classe de conteúdo *Prontuario*, com os atributos configurados com os valores enviados na requisição. Estes dados deveriam ser exibidos em uma página de resposta da aplicação. Estas requisições foram enviadas tanto via *browsers* (Mozilla Firefox e Internet Explorer) a partir de um computador pessoal (*desktop*), quanto a partir de um dispositivo móvel (via emulador do celular Nokia modelo N97).

Todas as requisições enviadas foram capturadas pelo *pointcut captureRequest* do aspecto CAS. No caso das requisições enviadas via *browsers* em computadores *desktop*, o *advice* associado configurou o atributo *targetPlatform* do aspecto CAS com uma instância da classe *DesktopTargetPlatform*, já nas requisições enviadas pelo *browser* do dispositivo móvel, a classe instanciada foi a *MobileDeviceTargetPlatform*. Isto demonstrou o correto funcionamento do *pointcut* e que as configurações realizadas em *AppServiceConfig* foram corretamente carregadas pelo aspecto CAS.

Por fim, os objetos *Prontuario* retornados pelo método *consultarProntuario* foram capturados pelo *pointcut captureResponse* do aspecto CAS em todas as respostas da aplicação. O processo de adaptação executou e, para as respostas a requisições efetuadas pelos *browsers desktop*, o objeto foi retornado sem adaptações. Nas respostas às requisições efetuadas a partir do *browser* em dispositivo móvel, o objeto sofreu a execução da *MobileDeviceAdaptationStrategy* e seus *LogicalProcedures* associados, sendo retornado com adaptações. Os resultados podem ser vistos na figura 8.

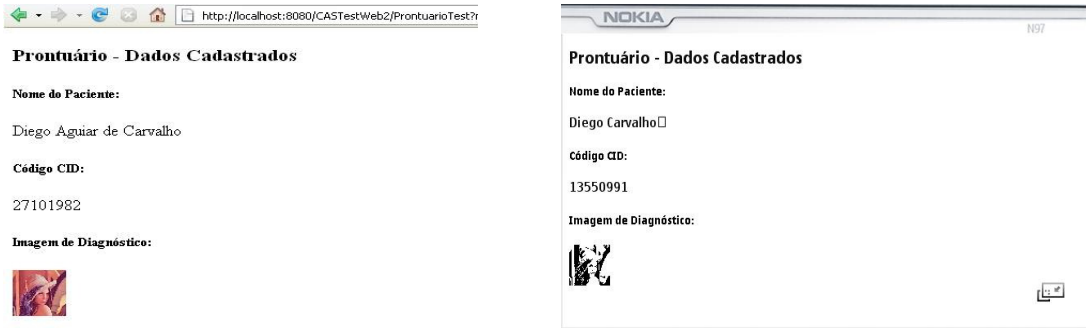

Figura 8. Resultado da execução do Serviço de Adaptação de Conteúdo sobre requisições à aplicação enviadas, respectivamente, via browser Firefox e via browser do dispositivo móvel Nokia N97.

5. Conclusões e Trabalhos Futuros

O estudo de caso realizado, embora ainda preliminar, permitiu verificar que os objetivos estabelecidos para este trabalho são viáveis e podem ser atingidos em plenitude. As principais observações sobre o resultado do experimento são: *(i)* o processo proposto foi aplicado em todas as suas etapas, demonstrando-se adequado para uma correta implementação da característica de adaptação de conteúdo a uma aplicação multiplataforma; *(ii)* a API proposta atendeu ao processo, todavia, considerando-se o grau de simplicidade da aplicação experimental, demandou esforço considerável de implementação; *(iii)* o serviço atuou de forma não intrusiva, não exigindo refatorações no código das classes da aplicação, o que permite considerar adequado o uso da tecnologia de orientação a aspectos em sua implementação.

Como trabalho futuro desta pesquisa, pretende-se evoluir a aplicação experimental e realizar novo estudo de caso, bem como aplicar estudo de caso sobre aplicação já implementada (legada), de domínio de negócio distinto. Pretende-se ainda, analisar o projeto *aopmetrics* [2], que provê uma ferramenta de métricas comuns para programação orientada a objetos e orientada a aspectos, e selecionar entre suas métricas quantitativas as mais adequadas aos objetivos da pesquisa, aplicando-as nos estudos de caso e na implementação do serviço de adaptação de conteúdo.

6. Referências

[1] ABOWD, G. D. et al. **Towards a Better Understanding of Context and Context-Awareness**. In Proceedings of the 1st international Symposium on Handheld and Ubiquitous Computing, Karlsruhe, Germany. 1999.

[2] AOP Metrics 2009. Disponível em < http://aopmetrics.tigris.org> Acesso em: 16 mar. 2009.

[3] BUSCHMANN, F., et al. **Pattern-Oriented Software Architecture: A System of Patterns**. John Wiley & Sons, 1996.

[4] CÁMARA, J., SALAUN, G., CANAL, C. **On Run-time Behavioural Adaptation in Context-Aware Systems**. 1st Workshop on Model-driven Software Adaptation at ECOOP 2007. 2007.

[5] CHAARI T., LAFOREST F., CELENTANO A. **Adaptation in Context-aware Pervasive Information Systems: The Secas Project**. Journal of Pervasive Computing and Communications, V.2, No.2, 2006.

[6] DINIZ, J. R. B. UbiDoctor: **Arquitetura de Serviços para Gerenciamento de Sessão e Adaptação de Conteúdo em Ambientes de Medicina Ubíqua**. 2008. Tese de Doutorado - Universidade Federal de Pernambuco, UFPE, Brasil.

[7] Eclipse 2009. Disponível em < http://www.eclipse.org/aspectj/doc/released/adk15notebook/annotations.html#annotations-inJava5> Acesso em: 16 mar. 2009.

[8] KICZALES G., et al. **Getting Started with AspectJ**. Communications of the ACM, 44(10):59{65, October 2001.

[9] KICZALES, G., et al. **Aspect-oriented programming**. In Proceedings of European Conference on Object-Oriented Programming (ECOOP), Finlândia. 1997.

[10] LEI, Z., GEORGANAS, N.D. **Context-based media adaptation in pervasive computing**. In Proceddings of IEEE Canadian Conference on Eletrical and Computer (CCECE'01), Toronto, Canada. 2001.

[11] SANTOS, I. L. A. **Uma Abordagem baseada em Aspectos e Composição Dinâmica para a Construção de Aplicações Adaptativas Cientes ao Contexto**. 2008. Dissertação de Mestrado - Universidade Federal do Rio Grande do Norte, UFRN, Brasil.

[12] TRINTA, F. **Definindo e Provendo Serviços de Suporte a Jogos Multiusuário e Multiplataforma: Rumo à Pervasividade**. 2007. Tese de Doutorado - Universidade Federal de Pernambuco, UFPE, Brasil.