# Implementação de uma infra-estrutura para controle de versões e adaptação de páginas Web

**Luiz Carlos de Freitas Santos Júnior, Rodrigo Giacomini Moro, Carlos Alberto Heuser**

Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil

lsantos@inf.ufrgs.br, rodrigo@urisantiago.br, heuser@inf.ufrgs.br

***Resumo.*** *Técnicas oriundas da área de Gerência de Configuração de Software, como controle de versões e adaptação dinâmica de páginas Web (de forma transparente ao visitante do site, através de um processo automático de configuração) auxiliam na manutenção e na atualização de sites Web. Este trabalho apresenta uma ferramenta que utiliza comandos do protocolo WebDAV para promover atualizações nos recursos Web. Através da criação ou modificação de páginas Web, o processo de configuração de páginas dinâmicas pode ser configurado e rodar em um servidor WebDAV especial, que implementa um modelo de versões de páginas separando conteúdo e formatação em componentes distintos. Este processo de configuração seleciona automaticamente versões de componentes para gerar páginas Web adaptadas conforme o perfil do usuário. Adicionalmente, permite que a comunicação com o servidor seja mais amigável para o usuário, pois a utilização deste aplicativo não requer conhecimento de comandos do protocolo. Além disso, a ferramenta permite controlar o acesso ao servidor, pois implementa os conceitos de usuário e controle de acesso. Esta ferramenta representa a aplicação utilizada tanto pelo visitante, para ver páginas adaptadas, quanto pelo editor (projetista do site), que pode acessar o servidor e gerenciar o repositório de recursos.*

## 1. Introdução

Conforme os *sites* Web crescem em número de páginas, sua manutenção torna-se mais complicada. Assim, os administradores de *sites* necessitam de métodos e ferramentas que tornem sua manutenção mais organizada e automatizada. Entretanto, a criação de tais mecanismos é dificultada pelo formato das páginas Web (HTML), que mistura o conteúdo e a formatação da página em um mesmo arquivo. Uma solução usual para esse problema [KAY, 2000] é separar esses componentes da página em documentos XML (conteúdo) e folhas de estilo XSLT (formatação). Técnicas da área de Gerência de Configuração de Software [ZELLER; SNELTING, 1997], como o controle de versões e configurações, e a adaptação automática de páginas também podem ser utilizadas para auxiliar na manutenção de *sites*.

Controlar versões de hiperdocumentos é um requisito importante, conforme verificado por [HALASZ, 1988]. Para atender a essa necessidade, vários modelos de versões de hiperdocumentos foram propostos [BIELIKOVÁ; NÁVRAT, 1998], [DELISLE; SCHWARTZ, 1987], [NORONHA et. al, 1998], [CHIEN, 2002].

Em [MORO, 2003] é proposta uma infra-estrutura para um servidor Web que realiza controle de versões e suporta adaptação de páginas Web de forma transparente ao visitante. Para tanto, é projetado um modelo de versões de páginas que separa conteúdo e formatação em componentes distintos. É proposto um processo de configuração que é responsável pela geração de páginas dinâmicas, no que é suportado por informações presentes no modelo de versões. Os autores de páginas e o próprio servidor Web podem interferir nas escolhas do processo de configuração, fornecendo critérios de seleção de versões. Esses critérios guiam as escolhas do processo de configuração, pois representam características que as versões escolhidas devem (necessariamente ou preferencialmente) apresentar. É implementado um protótipo de servidor Web para validar a infra-estrutura de controle de versões e adaptação de páginas Web.

Este trabalho descreve o projeto e a implementação de uma ferramenta que gerencia e facilita a comunicação dos usuários com o servidor WebDAV protótipo implementado em [MORO, 2003]. A ferramenta foi denominada Software Gerenciador de Infra-Estrutura para Controle de Versões e Adaptação de Páginas Web (SGICVAPW). Na seção 2 encontra-se descrita, de forma sucinta, a proposta de infra-estrutura para controle de versões e adaptação de páginas Web. A seção 3 descreve a implementação e os módulos da ferramenta de comunicação SGICVAPW. Na seção 4 são apresentadas as considerações finais.

## 2. Proposta de Infra-Estrutura para Controle de Versões e Adaptação de Páginas Web

Conforme descrito em [MORO, 2003], nesta seção é apresentado o modelo de versões para suportar adaptação de páginas Web. A Figura 1, através de um diagrama de classes UML, mostra os principais conceitos propostos no modelo de versões. A seguir, a Figura 1 é explicada detalhadamente.

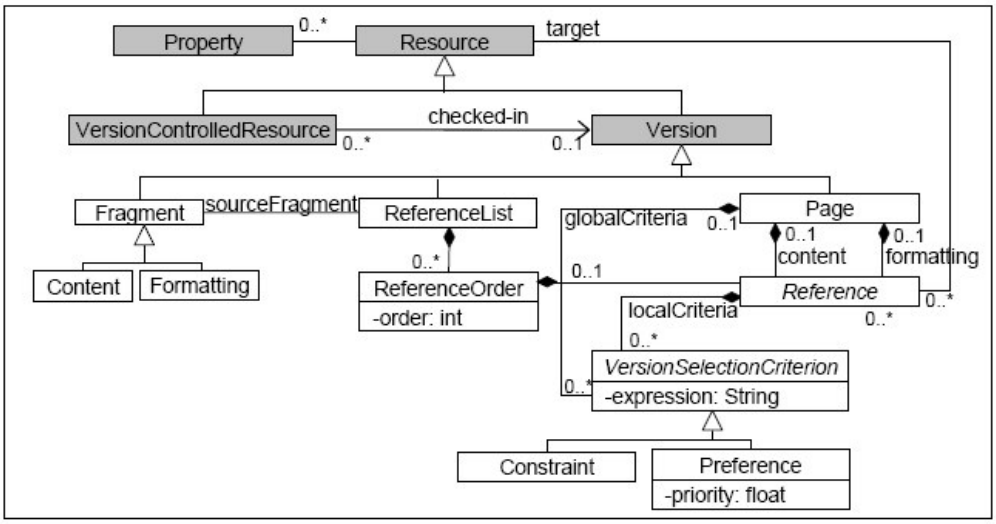


**Figura 1: Diagrama de classes UML do modelo simplificado**

O modelo proposto é uma extensão do modelo de dados do WebDAV. Os conceitos **Resource**, **Property**, **VersionControlledResource** e **Version** do WebDAV

são descritos com a cor de preenchimento cinza na Figura 1. Um **Resource** é um objeto de dados da rede ou serviço que pode ser identificado por um URI (*Uniform Resource Identifier*). Recursos podem ter informações descritivas (metadados) associadas, na forma de um par nome / valor, representado pela classe **Property**. Um **VersionControlledResource** é um recurso que tem todos os seus estados controlados pelo servidor, ou seja, quando o estado deste recurso é modificado, o estado anterior não é sobrescrito e o novo estado também é armazenado. Cada estado deste recurso é chamado de versão (conceito **Version**).

Esse modelo é baseado em referências que relacionam os componentes da página Web. A classe **Page** é um recurso que representa uma página Web para os desenvolvedores. A classe **Reference** representa um ponteiro de seu recurso origem a seu recurso destino (associação **target**). A página contém duas referências, uma para o conteúdo (**content**), que é um documento XML que armazena o conteúdo da página, e outra para a formatação (**formatting**) que é uma folha de estilo XSLT (ou algum outro formato de apresentação) que contém as regras de formatação da página.

Como conteúdo e formatação podem ser incluídos e reusados em outros documentos XML e folhas de estilo XSLT, definiu-se o conceito de fragmento, representado pela classe **Fragment**. Similarmente ao modelo de página, conteúdo e formatação também são separadamente mantidos em fragmentos. Adicionalmente, um fragmento pode ter referências para outros fragmentos de mesmo tipo (conteúdo ou formatação) para incluí-los. A associação **sourceFragment** representa uma lista contendo todas as referências de um fragmento origem de referência.

Em um hiperdocumento composto, no qual componentes podem conter versões, um *processo de configuração* seleciona exatamente uma versão de cada um desses componentes. Referências dinâmicas (apontam para recursos versionados) devem ser resolvidas, escolhendo uma versão de acordo com algum critério pré-definido. O comportamento *default* do processo de configuração é selecionar a versão mais recente para cada recurso versionado referido. Entretanto, esse comportamento não é sempre apropriado, pois o processo de configuração não tenta selecionar a melhor versão. Assim, deve haver algum meio de intervir nesse comportamento. Para resolver esse problema, os desenvolvedores podem associar critérios de seleção de versões (classe **VersionSelectionCriterion**) que guia o processo para a escolha de versões adequadas para a configuração da página, similarmente a uma consulta em um sistema de banco de dados. Usando critérios de seleção de versões, é possível escolher as versões mais adequadas até mesmo se novas versões são criadas posteriormente. As versões são escolhidas de forma que suas propriedades satisfaçam os critérios de seleção de versões. Além disso, dependendo do escopo, o critério pode ser local ou global (associações **localCriteria** e **globalCriteria**). O critério local é usado somente na solução de uma referência. Ao contrário, o critério global é usado para resolver todas as referências de uma requisição de página.

Um critério pode representar uma restrição ou uma preferência. Uma *restrição* especifica uma característica que a versão deve possuir. Isso significa que esse critério é imperativo, pois representa uma regra de consistência que deve ser satisfeita (classe **Constraint**). Uma *preferência* especifica características que são desejáveis na versão selecionada. Isso significa que esse é um critério opcional, pois ele especifica somente

uma preferência que pode ser satisfeita, mas que não é requerida obrigatoriamente. A classe **Preference** possui um atributo **priority** que indica sua importância; ele é usado pelo processo de configuração para estabelecer a ordem de avaliação das preferências.

Modelou-se também um recurso que armazena o critério separadamente dos fragmentos. Esse recurso é chamado de *lista de referências* (**ReferenceList**). Ele contém uma lista na qual os elementos (**ReferenceOrder**) indicam todas as referências de um fragmento fonte (associação *sourceFragment*). A ordem das referências (atributo *order* de **ReferenceOrder**) na lista de referências é a mesma do fragmento fonte. Para cada referência, a lista de referências armazena também seu critério de seleção da versão. Assim, um autor pode editar a lista de referências para especificar o critério local de seleção de versões para as referências do fragmento.

Um fragmento possui uma associação (chamada *lista de referências*) com um recurso versionado que é uma lista de referências. Quando um fragmento é criado ou atualizado, o servidor Web automaticamente extrai as referências do fragmento e usa-as para criar um recurso versionado para a lista de referências. A lista de referências e o fragmento devem estar associados pelas associações *referenceList* e *sourceFragment*.

A associação *referenceList* é usada pelo processo de configuração para localizar a versão atual (associação *checked-in*) da lista de referências. O processo necessita da lista de referência ordenada para obter os critérios (restrições e preferências) para a resolução de referências. A associação *sourceFragment* possibilita ao autor localizar o fragmento do qual as referências foram extraídas. Isso não pode ser feito através da associação *checked-in* porque essa associação é unidirecional, e não pode ser acessada pela versão atual da lista de referências.

## 3. Descrição da ferramenta

A ferramenta foi implementada na linguagem PHP e seu acesso pode ser feito através da Web, sendo, portanto, independente de sistema operacional. O aplicativo executa em um servidor Apache e acessa o servidor WebDAV protótipo via *sockets* da linguagem PHP, utilizando os seus comandos para adaptar páginas Web. A Figura 2 apresenta a arquitetura do sistema.

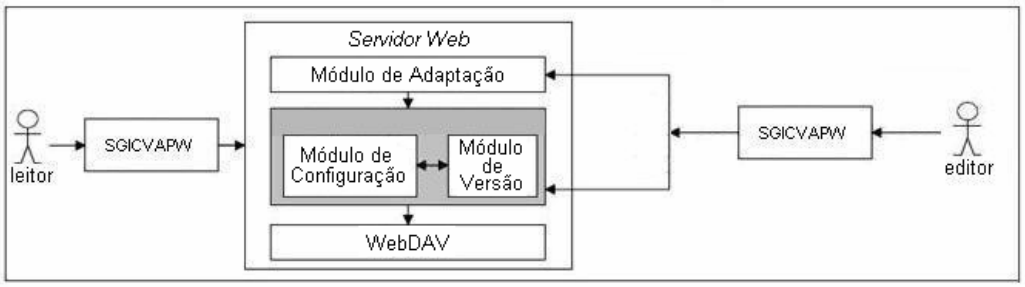


**Figura 2: Arquitetura do sistema**

Os usuários do SGICVAPW podem ser de dois tipos: administrador (editor) ou visitante (leitor) do *site*. Esta ferramenta representa a aplicação utilizada tanto pelo leitor, para ver páginas adaptadas, quanto pelo editor (projetista do *site*), que pode acessar o servidor e gerenciar o repositório de recursos.

As seguintes funções foram implementadas pelo aplicativo: Cadastrar Usuário, Criar Coleção, Criar Recurso, Criar Página (Descrição de Configuração), Preparar um Recurso para ser Modificado (CHECKOUT), Modificar Recurso, Modificar Página, Ver Propriedades, Adicionar Propriedades, Remover Propriedades, Criar Versão (CHECKIN), Solicitar Cabeçalho de Página, Solicitar Página Completa, Obter Informações do Recurso e Tornar Versão Folha. A seguir, algumas operações relevantes no contexto da adaptação de páginas Web são exemplificadas.

Para ilustrar a utilização da ferramenta, foi escolhido como exemplo um *site* de notícias de um jornal.

**Criar Coleção:** Gera uma nova coleção no sistema, onde é possível inserir recursos versionados (componentes de formatação e conteúdo, além de páginas com critérios de seleção de versões). A Figura 3 ilustra a criação de uma coleção no sistema.

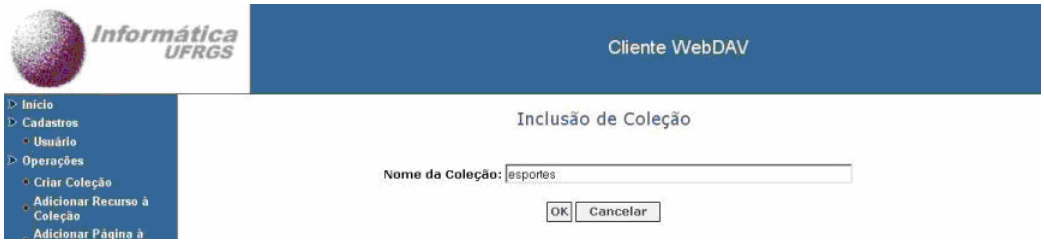


**Figura 3: Criar coleção**

**Adicionar Recurso à Coleção:** Insere em uma coleção já existente um recurso versionado de conteúdo ou formatação. A Figura 4 apresenta a tela desta operação.

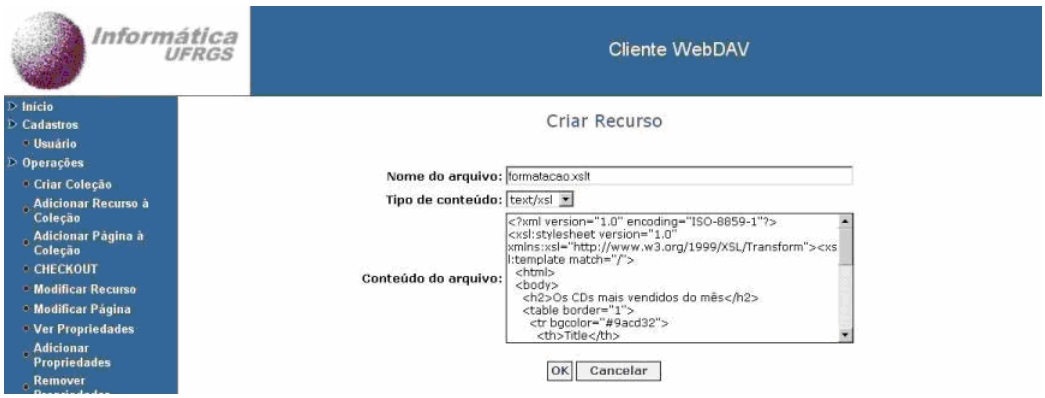


**Figura 4: Criar Recurso**

**Adicionar Página à Coleção:** Permite que seja inserida em uma coleção já existente uma descrição de configuração referenciando um componente de conteúdo e um componente de formatação, descrição esta que representa uma página Web para o servidor WebDAV. É aqui também que são inseridos os critérios globais de seleção de versões.

**Solicitar Página Completa:** Permite visualizar as páginas Web adaptadas no SGICVAPW. A Figura 5 apresenta uma página dinâmica selecionada pelo processo de configuração do servidor WebDAV.

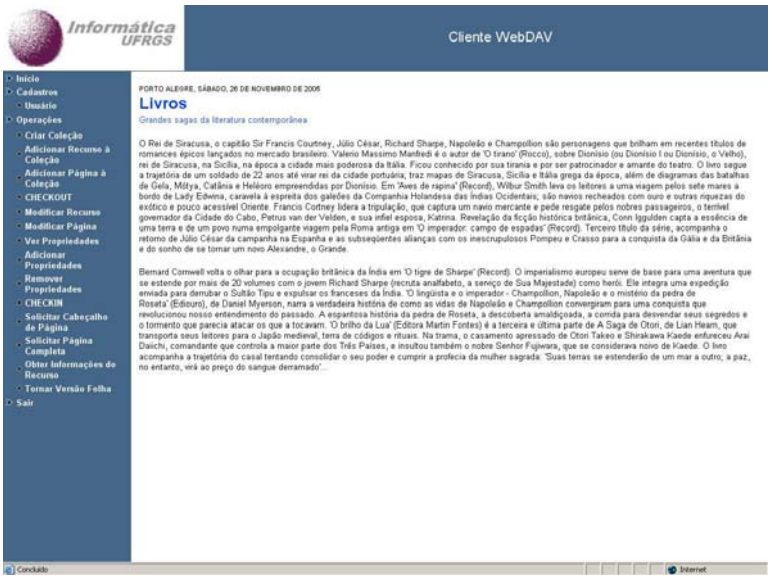


**Figura 5: Adicionar Página à Coleção**

## 4. Considerações finais

Observando o trabalho realizado é possível identificar aspectos que podem ser melhorados. Entre eles, dois se destacam: permitir a adição de imagens, áudio e vídeo ao criar e modificar recursos versionados; acelerar o processamento do servidor WebDAV na operação "Solicitar Página Completa", repassando as transformações XSLT para o *browser* e mantendo as páginas mais acessadas em *cache*.

## Referências Bibliográficas


BIELIKOVÁ, M.; NÁVRAT, P. Modelling Versioned Hypertext Documents. In: SYSTEM CONFIGURATION MANAGEMENT SYMPOSIUM, 8., 1998, Brussels, Belgium. Proceedings. . . Berlin: Springer-Verlag, 1998. p.188–197. (Lecture Notes in Computer Science, v.1439).

CHIEN, S.Y. et al. Efficient Complex Query Support for Multiversion XML Documents. In: CONFERENCE ON EXTENDING DATABASE TECHNOLOGY, EDBT, 8., 2002, Prague, Czech Republic. Advances in Database Technology: proceedings. . . Berlin: Springer-Verlag, 2002. p.161–178. (Lecture Notes in Computer Science, v.2287).

DELISLE, N.; SCHWARTZ, M. Context - a Partitioning Concept for Hypertext Enviroment. Transactions on Office Information Systems, [S.l.], v.15, n.2, p.168–186, Apr. 1987.

HALASZ, F. G. Reflections on Notecards: seven issues for the next generation of hypermedia systems. Communications of the ACM, New York, v.31, n.7, p.836–852, July 1988.

KAY, M. XSLT Programmer's Reference. UK: Wrox Press, 2000. 778p.

MORO, R. G. Uma infra-estrutura para controle de versões e adaptação de páginas Web. 2003. 78 f. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

NORONHA, M. A.; GOLENDZINER, L. G.; SANTOS, C. S. dos. Extending a Structured Document Model with Version Control. In: INTERNATIONAL DATABASE ENGINEERING AND APPLICATIONS SYMPOSIUM, IDEAS, 1998, Cardiff, Wales, U.K. Proceedings. . . Los Alamitos: IEEE Computer Society, 1998. p.234–242.

ZELLER, A.; SNELTING, G. Unified Versioning Through Feature Logic. ACM Transactions on Software Engineering and Methodology. New York: TOSEM, v.6, n.4, p.398–441, Oct. 1997.