UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

ALEXANDRE LUÍS FRANCO

Análise do Padrão W3C / XForms 1.0

Trabalho de Conclusão de Mestrado
apresentado como requisito parcial
para obtenção do grau de
Mestre em Sistemas de Informação

Prof. Dr. Carlos Alberto Heuser
Orientador

Porto Alegre, janeiro de 2004.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Franco, Alexandre Luís

Análise do Padrão W3C / XForms 1.0 / Alexandre Luís Franco – Porto Alegre: Programa de Pós-Graduação em Computação, 2004.

103f.: il.

Trabalho de Conclusão (Mestrado)-Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR-RS, 2004. Orientador: Carlos Alberto Heuser.

1. XForms. 2. XML. I. Heuser, Carlos Alberto. II. Título

AGRADECIMENTOS

À Estela, pelo apoio durante a execução deste trabalho.
Aos meus Pais Cláudio e Ilca, pela força e compreensão.
Aos meu tios José e Maria Estela, pela inspiração.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

CPF	Cadastro de Pessoa Física
CSS	Cascade Style Sheets
DOM	Document Object Model
DTD	Document Type Definition
FDF	Forms Data Format
HTML	Hypertext Markup Language
J2EE	Java 2 Enterprise Edition
JSP	Java Server Pages
MVC	Modelo – Visão - Controlador
PDF	Portable Document Format
PFF	Portable Form File
RFC	Request for Comments
RG	Registro Geral
URI	Universal Resource Identifier
VoiceML	Voice Markup Language
W3C	World Wide Web Consortium
WAP	Wireless Access Protocol
WML	Wireless Markup Language
WYSIWYG	What You See is What You Get
WWW	World Wide Web
XHTML	Extensible Hypertext Markup Language
XML	Extensible Markup Language
XPATH	XML Path Language
XSL	Extensible Stylesheet Language
XSLT	Extensible Stylesheet Language Transformation

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

O uso de formulários é uma das principais formas de interação existentes na Internet. Desde a versão 2.0 do HTML, entretanto, pouco foi melhorado no modelo de formulários proposto. Ao mesmo tempo, as necessidades dos desenvolvedores e os requisitos dos usuários cresceram dramaticamente.

O W3C apontou uma resposta para as necessidades levantadas, o padrão XForms. O padrão XForms visa substituir o modelo de formulários definido no HTML por um modelo que separa o propósito da apresentação, adicionando, desta forma, a característica de independência de plataforma.

A proposta deste trabalho é analisar o padrão XForms em relação à utilização de formulários HTML tradicionais, e à outras soluções existentes para automação de formulários na Internet, utilizando para isto uma aplicação piloto que procure utilizar alguns dos principais recursos disponíveis no padrão.

Os pontos fortes, pontos fracos, dificuldades e lições aprendidas capturadas durante o desenvolvimento da aplicação piloto formam uma base de conhecimento apresentada neste trabalho.

Palavras-chave: XForms, HTML, Internet, XML, XSLT, XML Schema, XML Events.

Analysis of the W3C XForms 1.0 Standard

ABSTRACT

The use of forms is one of the main interaction ways found on the Internet. However, since HTML version 2.0, a few improvements were made in the proposed forms model, while developer's needs and user requirements rose dramatically.

The W3C group answered those needs, with the XForms standard. The XForms standard aims to replace the forms model defined in HTML by a model that separates the purpose from the presentation on a form, thus adding the platform independence characteristic.

The purpose of this work is to analyze the XForms standard, related to traditional HTML forms, and to other existing web forms automation solutions. A pilot application was used for this analysis, using some of the characteristics available in the standard.

The strengths, weaknesses, difficulties and lessons learnt collected throughout the pilot application development are part of a knowledge base presented in this work.

Keywords: XForms, HTML, Internet, XML, XSLT, XML Schema, XML Events.

1 INTRODUÇÃO

Formulários são usados praticamente todos os dias, de uma forma ou de outra. Comércio eletrônico, mecanismos de busca, pesquisa, praticamente toda a interação Web é feita através de algum tipo de formulário (DUBINKO, 2002).

Além disso, o desenvolvimento de aplicações que utilizem formulários HTML é uma das atividades mais dispendiosas em projetos de software, se comparados a outras tecnologias. A obtenção de interfaces com o usuário leves e, ao mesmo tempo com alta usabilidade, requer o uso de artifícios complexos.

Diferentes aspectos são mantidos em um único documento que especifica um formulário HTML: layout e design, dados coletados e comportamento. Em grandes projetos, equipes distintas trabalham sobre cada um dos aspectos: web designers, analistas de aplicação / negócio e desenvolvedores de linguagens de script são um exemplo.

Visando endereçar estes e outros problemas, a automação de formulários tornou-se objeto de pesquisa, com o objetivo de simplificar o desenvolvimento de formulários e aumentar a produtividade do desenvolvedor. A resposta do W3C para esta necessidade é o padrão XForms 1.0, atualmente uma recomendação do W3C, disponível para revisão pública (DUBINKO et al., 2002).

Com o estudo sobre o padrão espera-se compreender melhor o problema atualmente existente com formulários HTML, e suas tentativas de automação.

O objetivo do presente trabalho é analisar o padrão XForms, utilizando para isso uma aplicação complexa real a ser definida. A aplicação deverá prover a visualização / preenchimento de formulários em múltiplas etapas, capacidade de suportar campos relacionados (por exemplo, caixas de seleção de País / Estado / Município) e internacionalização, entre outros.

Foi escolhida uma das implementações atualmente existentes para a execução da aplicação piloto. Entre elas, destacam-se o X-Simles (2003); Open XML Framework (OXF), da Orbeon (2003); o XML Forms Package da IBM (2003); o Xero, da "Type.a" software (2003); e o Chicoon, da Apache / IBM (BRANNAN, 2002).

O trabalho apresenta uma analise comparativa da utilização do padrão em relação à utilização de formulários HTML quanto aos seguintes aspectos:

- **Mescla de diferentes componentes**: formulários HTML são resultantes de uma mistura de componentes modelo e visão. O padrão XForms propõe uma separação lógica desses dois componentes. Este comportamento será verificado em situações reais, onde existam restrições de tempo e recursos, e os resultados serão comparados a formulários HTML padrões;

- **Dependência de Dispositivos**: formulários HTML originalmente foram definidos tendo-se em mente a independência de plataformas de hardware e software. Em um determinado momento da história, as restrições impostas pela independência de plataforma foram substituídas por interfaces mais complexas, dependentes de plataforma. O padrão XForms foi inicialmente projetado visando a independência total de dispositivos, assim como os formulários HTML. Em situações reais, com restrições de tempo e recursos, e com a exigência de alta qualidade, como o padrão XForms se comportará?
- **Arquitetura de duas camadas**: formulários HTML são desenvolvidos de forma a que o processamento encerre no lado servidor, em uma arquitetura tipicamente de duas camadas. Workflows muitas vezes são bem mais complexos do que isso, o que pode muitas vezes desencorajar o uso dessa tecnologia. O padrão XForms, por outro lado, foi criado visando atender demandas de preenchimento de formulários em múltiplas etapas, por diferentes pessoas. Será verificado o quão efetivo é o padrão em relação a esta restrição, comparando-se o resultado obtido com formulários HTML.
- **Dependência de linguagens de script**: Formulários HTML dependem basicamente de scripts para realizar cálculos e validações. O padrão XForms, por outro lado, define diversas operações para este fim, evitando a intervenção do autor para desenvolvimento com scripts.
- **Repetitividade e Retrabalho**: Desenvolvedores de aplicações baseadas em formulários HTML estão acostumados a usar "bibliotecas" de código comuns, muitos deles são copiados, colados e adaptados para uma determinada situação. O padrão XForms tende a reduzir ou eliminar este problema, ao propor uma arquitetura para formulários bem mais consistente. Será verificado o comportamento do padrão XForms quanto à repetitividade e retrabalho na aplicação piloto, e comparado em relação ao desenvolvimento usando formulários HTML
- **Dificuldade de Inicialização de Formulários**: A inicialização de formulários HTML tipicamente exige a construção de uma nova instância do formulário, ao contrário do padrão XForms onde bastaria se atribuir uma nova fonte de dados. Será verificada a solução do problema para a aplicação piloto, como se comporta em relação a formulários HTML.
- **Formato de Codificação**: Formulários HTML baseiam-se em capturar e enviar pares de nome / valor, enquanto o padrão XForms utiliza documentos XML para a estrutura de dados. A efetividade da solução proposta pelo padrão XForms será verificada, em relação à definição original dos formulários HTML.

O trabalho está organizado como segue:

A seção "Formulários HTML" apresenta brevemente os formulários HTML, com suas limitações e problemas conhecidos, para melhor compreender os fatores que levaram o W3C a propor um novo padrão para a geração de formulários.

A seção "Técnicas e Ferramentas para Desenvolvimento de Formulários HTML" apresenta algumas técnicas baseadas ou não em XML, bem como o padrão de projeto

modelo-visão-controlador (MVC). Uma análise do resultado destas soluções em relação aos problemas e limitações dos formulários HTML será apresentada.

A seção “O Padrão XForms 1.0” descreve o padrão XForms, e algumas implementações já existentes. Uma análise comparativa em relação às soluções anteriores será apresentada.

A seção “Definição das Aplicações Piloto” apresenta as aplicações que foram desenvolvidas utilizando o padrão, seguido de uma análise dos resultados obtidos em relação aos problemas e limitações encontrados em formulários HTML e aos objetivos – chave do padrão XForms.

A seção “Verificação dos Resultados Obtidos da Aplicação Piloto” descreve os pontos fortes, pontos fracos, lições aprendidas e dificuldades encontradas durante o desenvolvimento das aplicações piloto. Além disso, uma verificação quanto aos objetivos – chave do padrão XForms é apresentada, assim como uma verificação quanto aos problemas encontrados em formulários HTML.

A contribuição deste trabalho é uma base de conhecimento com os pontos fortes, pontos fracos, dificuldades e lições aprendidas que será coletada durante a utilização do padrão em uma aplicação piloto real e complexa. Eventualmente poderão surgir pontos que o padrão não contempla, os quais serão documentados.

2 FORMULÁRIOS HTML

Neste capítulo é feita uma revisão da origem dos formulários em aplicações na Internet, suas limitações e problemas conhecidos. Não serão descritos os elementos para definição de formulários HTML. Para tanto, recomenda-se a leitura do capítulo de formulários da RFC 1866 (BERNERS-LEE; CONNOLLY, 1995).

2.1 Um Breve Histórico

Em 1993 Daniel Connolly e Tim Berners-Lee apresentaram uma proposta para a versão 2.0 do HTML (Hypertext Markup Language), em um memorando que, entre outras coisas, introduziu elementos de formulários para entrada de informações pelo usuário (CONNOLLY; BERNERS-LEE, 1993). Os elementos de formulário são embutidos dentro de parágrafos, tabelas e outros elementos visuais.

Como resultado do workshop WWW em Julho de 1993, Dave Ragget (1993) esboçou o que seria conhecido como HTML+, numa tentativa de obter consenso antes da submissão formal de um rascunho para RFC. Nesta especificação, os formulários HTML foram detalhadamente descritos e exemplificados.

Em 1995 Tim Berners-Lee e Daniel Connolly finalizam a RFC 1866, que apresenta formalmente a versão 2.0 do HTML, com a definição de formulários, elementos de formulários e formas para a submissão de formulários.

Desde então a especificação de formulários HTML foi sendo enriquecida pelas novas versões do padrão HTML, definidas pelo W3C, e, mais recentemente, pelo padrão XHTML. A estrutura básica definida por Connolly e Berners-Lee (1993), entretanto, continua inalterada.

O padrão XHMTL 2.0 (W3C, 2003), atualmente um rascunho do W3C, está introduzindo uma mudança considerável, ao substituir formulários HTML for XForms. O XHTML 2.0 ainda está em desenvolvimento, mas está clara a importância do XForms no futuro do XHTML (DUBINKO, 2003).

2.2 Limitações e Problemas de Formulários HTML

Serão abordados nesta seção os diferentes pontos fracos e problemas encontrados no desenvolvimento de aplicações que utilizem formulários HTML.

2.2.1 Mescla de diferentes componentes: modelo, visão e controlador

A especificação HTML para formulários atualmente mescla diferentes módulos em um mesmo documento de trabalho:

- Modelo: são os dados manipulados pelo formulário, e as regras de negócio
- Visão: é a interface com o usuário.
- Controlador: é a ligação entre os módulos visão e modelo

Na Figura 2.1 podemos observar um formulário HTML bastante simples, que será utilizado para ilustrar os diferentes módulos existentes.

```
<html>
<head>
<title>Untitled Document</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<script language="JavaScript">
<!--
function goAhead() {
    if (document.frm.txtUsuario.value.length == 0) {
        alert('O campo usuário é obrigatório');
        return false;
    }
    document.frm.submit();
}
-->
</script>
</head>

<body bgcolor="#FFFFFF">
<form name="frm" method="post" action="http://www.mycompany.com/goThere.do">
  <p>Usu&aacute;rio:
    <input type="text" name="txtUsuario" value="meuusuario">
  </p>
  <p>Senha:
    <input type="password" name="txtSenha" value="minhasenha">
  </p>
  <p>
    <input type="button" name="Submit" value="Valida" onClick="goAhead();">
  </p>
</form>
<p> </p>
</body>
</html>
```

Figura 2.1: Exemplo de Formulário HTML

O resultado da apresentação deste documento em um navegador Internet pode ser visto na Figura 2.2.

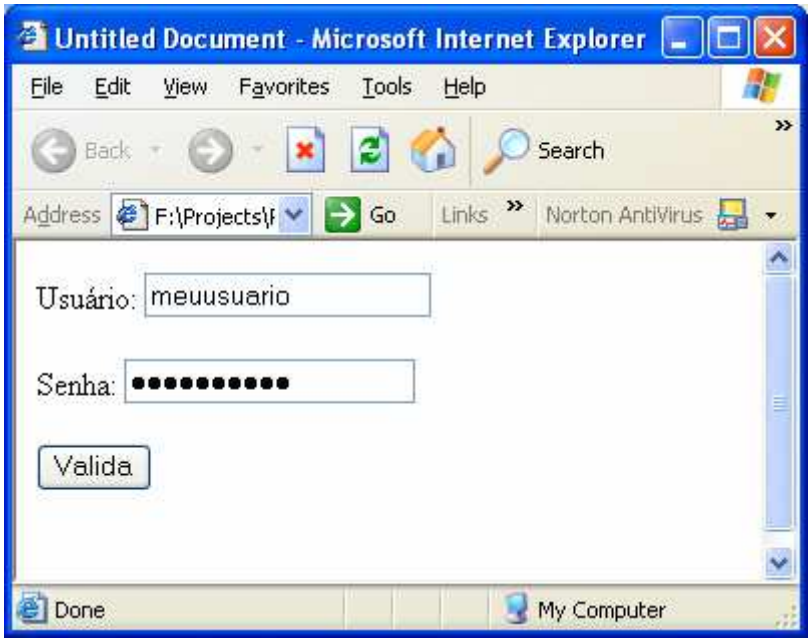


Figura 2.2: Resultado da Visualização no Navegador Internet Explorer

O modelo é representado pelo par nome / valor, definido por cada um dos elementos do formulário. No exemplo citado, o modelo é definido pela estrutura de dados exibida na Figura 2.3

```
txtUsuario=meuusuario
txtSenha=minhasenha
```

Figura 2.3: Modelo de pares nome / valor

Nota-se que a definição da estrutura do modelo está embutida no documento que define o formulário HTML.

A visão, que é a interface com o usuário, está incorporada na estrutura do documento principal. A definição de parágrafos (elementos “<p>”), cor de fundo, espaços em branco e nomes de campos são tipicamente funções da camada de apresentação, e podem ser facilmente identificadas no nosso exemplo.

O controlador é responsável por mesclar os componentes modelo e visão para gerar um formulário HTML. O resultado final é um documento que embute as três diferentes perspectivas em um único documento.

Em aplicações de pequeno porte essa mistura de conceitos pode não representar maiores problemas. Muitas vezes a equipe que trabalha nos diferentes módulos da aplicação é a mesma, e a divisão fica praticamente transparente.

Em aplicações maiores, entretanto, tipicamente serão utilizadas diversas equipes com diferentes habilidades em cada uma das tarefas.

Um exemplo típico em grandes aplicações, que utilizem formulários HTML, é a *visão* ser definida por uma equipe de *web designers*, enquanto que o *controlador* é feito por uma equipe de programadores de linguagens de script. A linha de tempo teoricamente não deveria coincidir, ou seja, programadores de scripts não deveriam trabalhar ao mesmo tempo em que *web designers* definem o layout da aplicação.

Tipicamente o *web designer* encerra seu trabalho entregando um protótipo para a próxima equipe, responsável por definir o *controlador* do formulário.

Uma vez iniciado o trabalho do *controlador*, uma modificação na *visão* (layout) pode representar a perda de todo o trabalho já desenvolvido no *controlador*, já que o entrelaçamento e interdependência entre módulos distintos em formulários HTML é bastante grande.

2.2.2 Dependência de Dispositivos

O objetivo original do HTML era prover uma linguagem de marcação independente de dispositivo, baseada na semântica do documento (BUTLER et al., 2002). O documento especificava elementos como cabeçalhos, parágrafos e listas sem especificar detalhes sobre como um determinado dispositivo implementaria visualmente o elemento.

Com o aumento da complexidade das aplicações, e com a exigência cada vez maior por interfaces com o usuário amigáveis e diferenciadas, os programadores e *web designers* começaram a reduzir a abrangência e focaram em algumas implementações para determinados dispositivos.

O *layout* de formulário começou a ser desenvolvido visando um determinado dispositivo ou resolução de tela. A tentativa de visualizar um documento em dispositivos ou resoluções distintos do original acaba por trazer resultados inesperados.

Diferenças de implementação até mesmo entre fabricantes de navegadores acarretaram no desenvolvimento de aplicações não somente dependentes de dispositivos como também dependentes de fabricantes de navegadores. O objetivo de buscar um diferencial competitivo no mercado possibilitou o desenvolvimento de aplicações com interfaces gráficas extremamente complexas, mas, ao mesmo tempo, a portabilidade ficou seriamente comprometida.

Para evitar este problema, algumas aplicações implementam de forma redundante diferentes porções do mesmo código, para atender a diferentes plataformas / navegadores existentes no mercado. Outras aplicações procuram transferir na medida do possível a complexidade para o lado servidor, de tarefas que tipicamente deveriam ser executadas no lado cliente.

2.2.3 Arquitetura duas Camadas

Um problema inerente ao uso de HTML para formulários é a arquitetura de duas camadas (cliente / servidor), com o processamento encerrando no lado servidor (DUBINKO, 2003). Diversas aplicações exigem um processamento mais complexo do que esse. Workflows, por exemplo, exigem muitas vezes diferentes etapas de preenchimento de formulários até o seu encerramento. Pequenas porções do formulário são preenchidas por pessoas diferentes em tempos diferentes.

O design de aplicações que implementem workflows em HTML envolve a reinterpretação dos dados em cada etapa do fluxo de trabalho. Para cada etapa, os dados precisam ser formatados em um novo formulário HTML, de forma que o ator possa dar prosseguimento ao fluxo estabelecido.

2.2.4 Dependência de Linguagens de Scripts

A implementação do componente controlador em formulários HTML é realizada através de scripts. Formulários HTML dependem essencialmente de scripts para a realização de tarefas comuns como o gerenciamento de campos obrigatórios, validação de campos, cálculos, exibição de mensagens de erro e gerenciamento de layout dinâmico (DUBINKO, 2003).

2.2.5 Repetitividade e Retrabalho

Aplicações que utilizem formulários HTML seguem tipicamente um padrão de comportamento: busca de dados para edição, montagem e apresentação do formulário, preenchimento do formulário, validação do formulário, aplicação de regras de negócio e armazenamento em alguma estrutura de dados.

Em aplicações um pouco mais complexas estas etapas precisam ser implementadas em arquiteturas escaláveis, que suportem, por exemplo, o grande público da Internet. Poderia-se ainda incluir uma etapa de validação de segurança da informação, responsável pela verificação de entradas de dados maliciosas (injeção de scripts, sobrecarga de campos).

A execução destas tarefas em diferentes formulários acaba se tornando repetitiva. Muitas vezes o desenvolvedor acaba utilizando o recurso de copiar e colar conteúdo de rotinas de outras aplicações.

Um exemplo é a validação de campos de formulários, que exige a implementação, via scripts, de algum mecanismo que valide o tipo de dados do elemento do formulário. Tipicamente estas rotinas são as mesmas, e são replicadas para toda e qualquer aplicação que utilize formulários HTML.

A conseqüência do retrabalho é a necessidade de mapeamento de onde as rotinas duplicadas estão sendo utilizadas. Em uma necessidade de atualização de uma das rotinas (por exemplo, verificação de entradas de dados maliciosas), é necessário replicar a atualização em todas as instâncias de formulários de todas as aplicações que a estão utilizando.

2.2.6 Dificuldade de Inicialização de Formulários

Dubinko (2003) destaca como um dos grandes problemas em formulários HTML a dificuldade de inicialização de dados, como normalmente acontece em sites web que lembram de usuários antigos e provêm a eles a cortesia de não ter que entrar informação repetidamente.

Cada campo do formulário possui uma forma única de definir dados iniciais. Isso é feito em conjunto com a montagem do documento HTML que contém o formulário. Tipicamente esta inicialização fica espalhada por todo o documento.

A transformação de um formulário em branco para um formulário preenchido exige que um novo formulário seja criado, ou que seja ajustado em diversos pontos. Essa necessidade exige bastante processamento, e pode se tornar um gargalo em servidores de alto volume (DUBINKO, 2003).

2.2.7 Formato de Codificação

Os dados capturados em formulários HTML seguem o padrão de par nome / valor. Na Figura 2.3 pode-se ver um exemplo de par nome / valor, no formulário hipotético de validação de usuários.

O formato nome / valor é adequado para pequenos formulários. Em aplicações complexas, entretanto, torna-se um problema já que a estrutura de dados é conseqüentemente bem mais complexa. Alguns artifícios acabam sendo necessários para lidar com tais aplicações, como a definição de nomes que embutem regras dentro dos mesmos. Na Figura 2.4 pode-se observar a especificação de um formulário HTML que espera o preenchimento dos cinco *hobbies* preferidos do usuário. A
Figura 2.5 mostra o resultado que pode ser visualizado no navegador Internet Explorer. Finalmente, o modelo de pares nomes / valores pode ser visto na Figura 2.6

```
<html>
<head>
<title>Untitled Document</title>
<meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1">
</head>

<body bgcolor="#FFFFFF">
<form method="post" action="">
  <p>Informe seus hobbies preferidos</p>
  <p>1.
    <input type="text" name="hobbie1">
    <br>
    2.
    <input type="text" name="hobbie2">
    <br>
    3.
    <input type="text" name="hobbie3">
    <br>
    4.
    <input type="text" name="hobbie4">
    <br>
    5.
    <input type="text" name="hobbie5">
  </p>
  <p>
    <input type="submit" name="Submit" value="Envia">
  </p>
  <p> </p>
</form>
</body>
</html>
```

Figura 2.4: Exemplo de formulário HTML com estrutura de dados não convencional

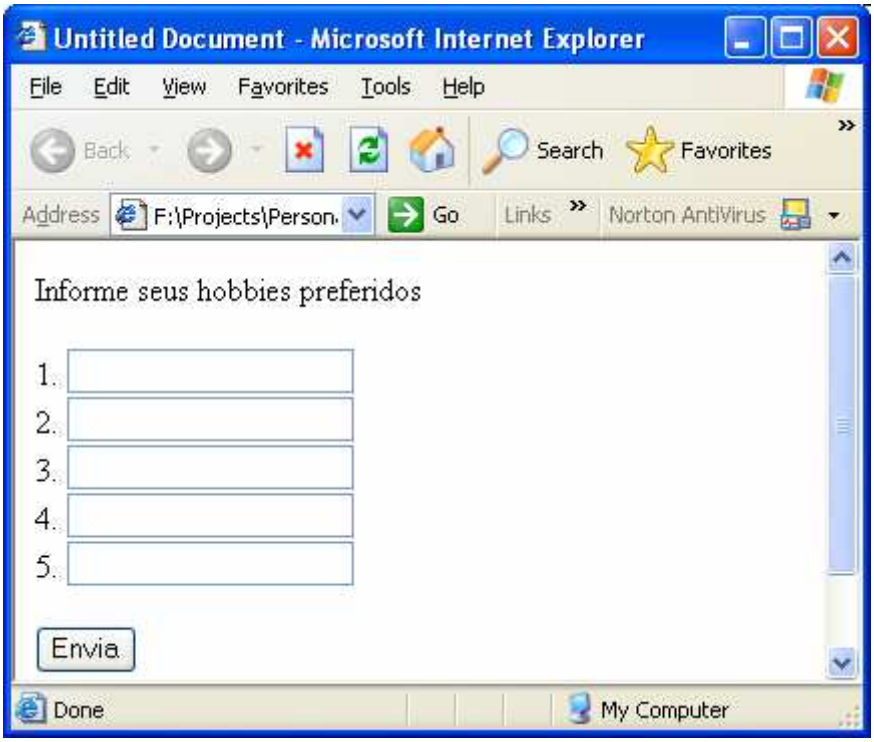


Figura 2.5: Apresentação de formulário HTML com estrutura não convencional

```
hobbie1=paddle
hobbie2=natação
hobbie3=tênis
hobbie4=cinema
hobbie5=teatro
```

Figura 2.6: Modelo de pares nome/valor para o formulário HTML não convencional

Pode-se perceber no exemplo da Figura 2.4 a inserção uma regra de formação para o nome dos cinco principais *hobbies* do usuário. A regra consiste em se adicionar um número seqüencial ao final do nome do campo.

O uso de uma outra estrutura de dados para a representação do modelo, como o XML, evitaria o uso de tais artifícios, tornando a aplicação mais clara e coerente.

3 TÉCNICAS E FERRAMENTAS PARA DESENVOLVIMENTO DE FORMULÁRIOS HTML

Para minimizar os problemas e limitações identificados no capítulo anterior, o processo de automação da criação de formulários HTML tornou-se o objetivo de diversos pesquisadores e fabricantes. A idéia era simplificar o desenvolvimento de aplicações que pudessem ser acessadas de qualquer lugar.

Neste capítulo serão abordadas algumas técnicas de automação da elaboração de formulários. Inicialmente será apresentada uma descrição do padrão de projeto modelo – visão – controlador, com benefícios que pode trazer na automação de formulários HTML. Posteriormente, ferramentas para o desenvolvimento de formulários não baseadas em XML serão discutidas. Por último, serão apresentadas algumas técnicas e ferramentas para o desenvolvimento de formulários que usam XML.

3.1 Arquitetura Modelo – Visão – Controlador (MVC)

Muitos problemas podem surgir quando as aplicações contêm uma mistura de código de acesso a dados, regras de negócio e código de apresentação (GOLDBERG, 1983). O alto acoplamento conseqüente desta abordagem dificulta a manutenção e o reuso.

A necessidade de uma aplicação ser executada em diferentes dispositivos, como um web browser, um dispositivo WAP, ou uma interface web service baseada em XML sugere a desvinculação da camada de apresentação do resto da aplicação (SUN, 2002).

A arquitetura modelo-visão-controlador (MVC) foi originada no Smalltalk (GOLDBERG, 1983) como um framework para o desenvolvimento de aplicações que trabalham com interfaces gráficas. Tipicamente essas aplicações possuem três componentes: o modelo, responsável pelo armazenamento dos dados específicos da aplicação; a visão, responsável pela apresentação do modelo em um dispositivo, como uma tela; e o controlador, responsável pelo mapeamento do modelo à visão (PREE, 1995).

O modelo representa os dados da organização e regras de negócio. É independente de representações específicas de saída (BUSCHMANN et al., 1996), bem como independente de comportamentos de entradas de dados. Mantém o estado da aplicação e, em muitos casos, é a representação lógica de um conceito no mundo real. Um exemplo de modelo é uma ficha de pessoa física, que contém campos para o armazenamento dos dados cadastrais da pessoa.

A visão mostra informações para o usuário, obtidas do modelo. Determina o formato como essas informações devem ser exibidas. É representada pela camada de interface

com o usuário. Podem existir múltiplas visões para um mesmo modelo (BUSCHMANN et al., 1996).

Por último, o controlador é a camada do meio no padrão MVC. Define o comportamento da aplicação, selecionando visões para apresentação e captura da interação com o usuário com as visões, roteando-as como comandos para o modelo. Em aplicações Web, as interações tipicamente aparecem como requisições http do tipo GET ou POST (SUN, 2002). O controlador é a cola que junta o modelo e a visão em um software aplicativo.

Um diagrama que ilustra os relacionamentos entre os três componentes pode ser visto na Figura 3.1.

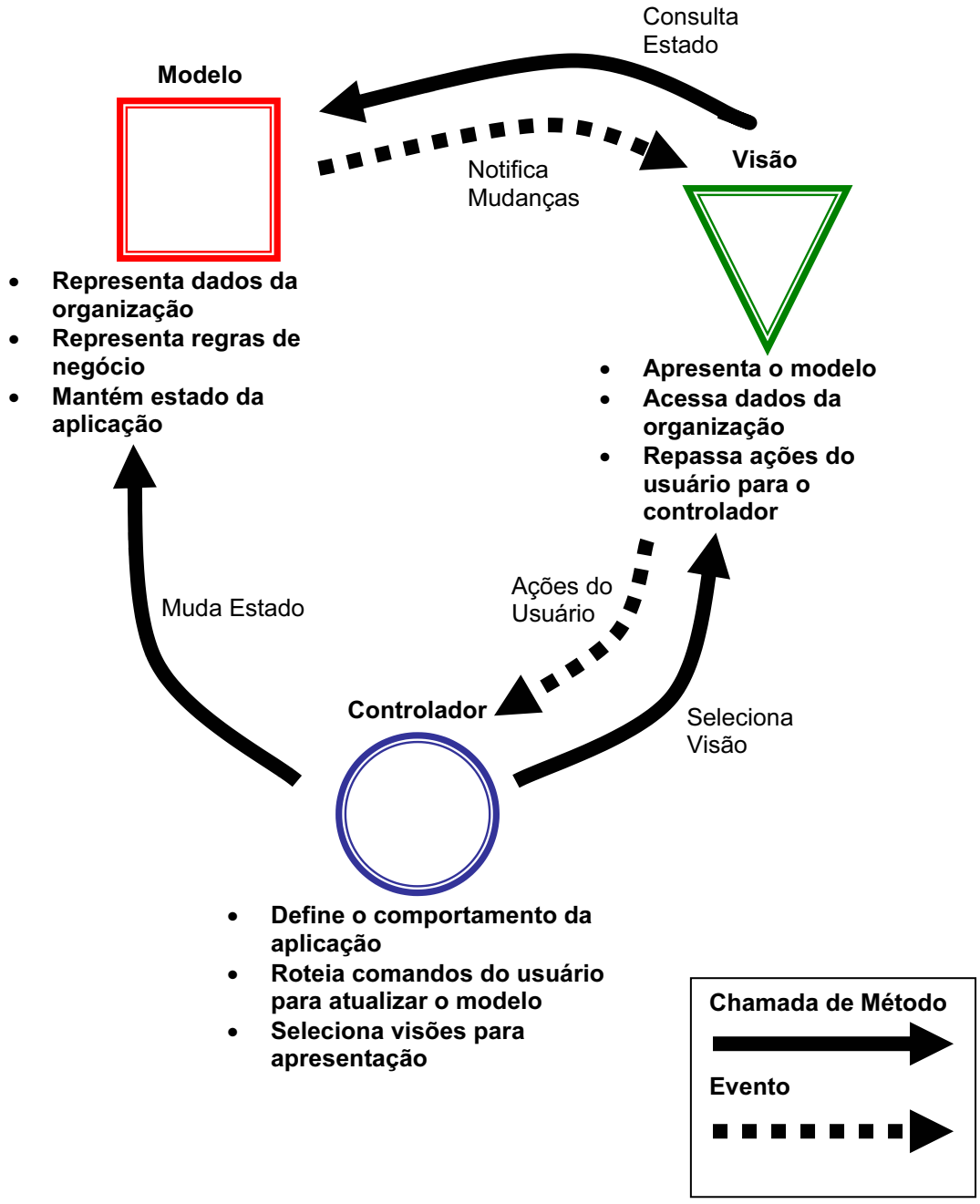


Figura 3.1: Esquema do Padrão MVC (ORACLE, 2003)

O benefício principal da separação destes três componentes é a facilidade de manutenção e entendimento da aplicação. Buschmann (1996) salienta a suscetibilidade à mudanças a que o componente visão está sujeito, devido à solicitações do cliente, ou até mesmo à mudanças de plataforma. Caso a interface com o usuário esteja fortemente amarrada com as funcionalidades básicas da aplicação, a manutenção e portabilidade serão comprometidas.

Além disso, os componentes pertencentes ao modelo tipicamente são facilmente reutilizáveis. Múltiplas visões podem usar o mesmo modelo. A longo prazo, as

aplicações tornam-se mais fáceis e mais rápidas de serem implementadas, com qualidade crescente.

Novos tipos de interfaces com o usuário são facilmente adicionadas, quando necessário. Supondo que uma determinada aplicação web precise ser portada para dispositivos WAP, basta o desenvolvimento de uma nova visão específica para este dispositivo.

Uma conseqüência negativa da utilização do MVC é a complexidade aumentada, fruto da separação dos módulos. Tipicamente o número de componentes (classes) da aplicação será bem maior.

3.2 Ferramentas de Desenvolvimento Não Baseadas em XML

As soluções não baseadas em XML foram as primeiras a surgirem com a intenção de automatizar a produção e gerenciamento de formulários. Entre elas, destacam-se a solução do Adobe eForms (ADOBE, 2001) e da Formatta EForms (1997).

Apesar dos componentes de visão e controlador não serem implementados em XML, nas duas ferramentas observa-se a possibilidade da implementação do modelo em XML.

Para cada uma das soluções propostas, será feita uma breve descrição seguida de uma análise quanto aos problemas e limitações identificados em formulários HTML.

3.2.1 O Software Adobe Acrobat eForms

O software Adobe Acrobat eForms é a solução da Adobe para formulários eletrônicos, baseado em documentos no formato Portable Document Format (PDF). O formato PDF é uma especificação disponível publicamente, que se tornou padrão de fato para a distribuição confiável de documentos online (ADOBE, 2001).

Segundo a Adobe (1993), os benefícios chave do software eForms são:

- Criar formulários eletrônicos que qualquer um, incluindo pessoas com problemas de visão, possa ler através de diversos dispositivos de hardware e software
- Minimizar o treinamento e cumprir as exigências da legislação, ao criar formulários eletrônicos que são visualmente idênticos aos papéis originais.
- Melhorar e simplificar o processamento de formulários com cálculos automáticos e validação de dados, assinatura eletrônica e integração com o banco de dados

Tipicamente as etapas de criação de um eForm são as seguintes:

1. **Definição do layout do formulário**. Com o uso de uma ferramenta de layout de página ou um software gráfico, pode-se desenhar o layout do formulário e transformá-lo em um documento PDF.
2. **Definição dos campos do formulário**. Com o uso do software Adobe Acrobat, adiciona-se campos para preenchimento pelo usuário. Cada campo deve ser unicamente identificado.

3. **Definição das ações e eventos**. Uma vez definido o layout do formulário e os campos, deve-se definir as operações que serão realizadas com os dados preenchidos. Exemplos de operações são enviar o formulário para alguém, assinar o formulário ou realizar uma cópia do formulário. Algumas operações já estão pré-definidas dentro da ferramenta, não necessitando nenhum esforço de programação pelo desenvolvedor. Ações mais complexas podem ser realizadas através do uso de linguagens de script, como o Javascript.

A submissão do formulário preenchido pode ser feita de diversas maneiras. Pode-se enviar os dados para um servidor Web, usando o método http POST. Os dados podem ainda ser armazenados em uma estrutura de dados chamada FDF, separados do documento PDF do formulário. O armazenamento dos dados preenchidos pode ser feito em XML, apesar de que a estrutura do formulário em si ser feita em PDF.

Vamos agora avaliar o software Adobe Acrobat eForms quanto a cada um dos problemas e limitações encontrados em formulários HTML:

a) Mescla de diferentes componentes: modelo, visão e controlador

O software Adobe Acrobat eForms exibe a possibilidade de manipular os dados submetidos pelo usuário em um documento separado, o FDF. Desta forma, pode-se dizer que há uma separação do modelo dos demais componentes da aplicação.

Os componentes de visão e controlador, entretanto, estão mesclados em um mesmo documento de trabalho. Inicialmente o componente da visão é definido, em uma ferramenta gráfica qualquer, e convertida para o formato PDF. Em seguida os campos são adicionados, e, por fim, seguido do comportamento embutido no formulário.

b) Dependência de dispositivos

O formato PDF é conhecido por sua independência de dispositivos e plataformas de software, existindo implementações de visualizadores em praticamente qualquer ambiente. A precisão do componente de visão é mantida em cada uma das plataformas implementadas.

c) Arquitetura de duas camadas

O software Adobe Acrobat eForms pode ser embutido em um ambiente de workflow chamado Adobe Approval, permitindo o preenchimento do formulário em múltiplas etapas.

d) Dependência de Linguagens de Scripts

Apesar de algumas funcionalidades do componente controlador do eForms estarem embutidas na própria ferramenta, a maior parte exige o uso de uma linguagem de script (Javascript) para a implementação das regras de negócio, validações e cálculos.

e) Repetitividade e Retrabalho

Algumas tarefas inerentes a manipulação de formulários HTML são automatizadas no uso do software Adobe Acrobat eForms. Um exemplo é a segurança da informação e

acesso, feita pelo Acrobat Approval. Entretanto, devido à dependência de scripts, alguns pontos ainda ficam sujeitos ao problema de repetitividade e retrabalho descrito para formulários HTML.

f) Dificuldade de Inicialização de Formulários

A possibilidade de separação do modelo em documentos FDF inibe a necessidade de instanciação de um novo formulário cada vez que se deseja abrir uma instância. Apenas o documento FDF é alterado.

g) Formato de Codificação

O software Adobe Acrobat eForms possibilita a separação do modelo em documentos FDF. O FDF por sua vez é uma estrutura de dados bastante simples, com pares de nome / valor.

Existe uma versão XML para o FDF, o XFDF, que possibilita o armazenamento do modelo em formato XML. Desta forma, as limitações do modelo composto por pares nome/valor são eliminadas.

3.2.2 O Software Formatta EForms

O software Formatta EForms é uma solução originalmente desenhada para endereçar requisitos do governo federal dos Estados Unidos, e companhias de todos os tamanhos (FORMATTA, 1997). Através de documentos do tipo *Portable Form Files* – PFF, um formato binário, os usuários podem preencher formulários de maneira amigável.

Segundo o fabricante, as principais características do software Formatta EForms são as seguintes (FORMATTA, 1997):

- Preenchimento simplificado. A interface com o usuário do software Formatta EForms é simples, apesar de possuir recursos poderosos.
- O autor do formulário é capaz de determinar as funcionalidades adicionais que serão utilizadas no formulário.
- Manutenção de dados e formulário juntos. É impossível associar arquivos de dados incorretos a um formulário
- Segurança. O software Formatta EForms pode ser criptografado para proteger os dados do usuário.
- Portabilidade. Os formulários podem ser preenchidos *online* ou *offline*; podem ser submetidos para servidores web via http, https, e-mail, anexos, SMTP ou FAX; podem ser materializados em códigos de barras bidimensionais para impressão e armazenamento físico;
- Integração de dados. Ações podem ser adicionadas para automatizar funções comuns, programar cálculos e validações, e até para submissão de formulários

Através de uma ferramenta WYSIWYG, o autor desenha o formulário, especifica campos e regras de negócios simples.

A verificação quanto aos problemas e limitações inerentes a formulários HTML é discutida a seguir.

a) Mescla de diferentes componentes: modelo, visão e controlador

O Software EForms da Formatta, por definição, incorpora o modelo junto aos componentes restantes do formulário. O fabricante alega motivos legais para isto, já que os dados e transações devem ser auditados e verificados.

b) Dependência de dispositivos

Os documentos PFF – Portable Form File são visualizados através do software Formatta Filler, uma ferramenta disponível para download. Atualmente esta ferramenta está disponível apenas para a plataforma Windows.

c) Arquitetura de duas camadas

O software Formatta EForms implementa o conceito de arquitetura em duas camadas, ou seja, o cliente preenche o formulário, transmite-o em algum formato (até mesmo em papel, com código de barras bidimensionais). Uma vez que o formulário tenha chegado ao servidor, este o decodifica e o armazena em algum banco de dados especificado pelo autor.

Entretanto, o software Formatta EForms possibilita o roteamento de formulários, permitindo desta forma a implementação de mecanismos de autorização / aprovação.

d) Dependência de Linguagens de Scripts

O software Formatta EForms possui embutido funções de validação e cálculo para a implementação do componente controlador no próprio formulário. Por outro lado isso pode representar uma limitação quando à capacidade de atender diferentes necessidades dos usuários.

e) Repetitividade e Retrabalho

A forma de trabalho definida pela ferramenta Formatta Designer, e as limitações impostas na implementação do componente controlador reduzem o retrabalho e a repetitividade inerentes a formulários HTML. Com funções predefinidas, o autor limita-se a utilizá-las na definição do formulário.

f) Dificuldade de Inicialização de Formulários

Os dados são parte integrante do formulário no software Formatta EForms. Os dados não são armazenados isoladamente, mas sim em conjunto com a especificação do formulário. A inicialização do formulário é, portanto, dependente da instância que se deseja visualizar.

g) Formato de Codificação

Os dados coletados em formulários no software Formatta EForms não se limitam a estrutura de dados de pares nome/valor. Pode-se especificar uma estrutura XML para o armazenamento do componente modelo, assim como na solução da Adobe.

3.3 Ferramentas e Técnicas de Desenvolvimento Baseadas em XML

A proximidade do Extensible Markup Language (W3C, 2003) ao HTML é inquestionável. O uso de transformações XSLT (W3C, 2003) aplicada a documentos XML para a geração de documentos HTML é amplamente utilizada comercialmente.

A transformação de documentos XML em formulários é uma conseqüência natural, implementada por diversos fabricantes. Além de transformações XSLT, o desenvolvedor pode optar por utilizar alguma outra técnica de varredura de documentos XML, como o Document Object Model (W3C, 2003)

Nesta seção vamos abordar algumas possíveis implementações que usam XML para a especificação de estruturas de formulários e analisar quanto aos problemas e limitações levantados em aplicações que utilizem formulários HTML.

3.3.1 Técnicas de Desenvolvimento Baseadas em XML / XSLT

Na solução que utiliza XML e XSLT para a geração de formulários, tipicamente o modelo é implementado em um documento XML, sendo os componentes de visão gerados a partir de transformações XSL (XSLT) sobre o modelo. Pode-se ter para um determinado modelo (documento XML) diversas visões resultantes de transformações geradas por diversos documentos XSLT.

Pode-se ver na Figura 3.2 um exemplo de documento XML que especifica o modelo de uma aplicação bastante simplificada. O modelo possui apenas três campos: cpf, nome e rg.

```
<?xml version="1.0" encoding="UTF-8"?>
<pessoa cpf="">
   <nome/>
   <rg/>
</pessoa>
```

Figura 3.2: Exemplo de Documento de Especificação de Modelo em XML

Na Figura 3.3 pode-se observar uma transformação XSLT que gera um formulário com campos textos para nome e RG.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <xsl:template match="/">

        <html>
        <head></head>
        <body>
        <form method=”post” action=”http://something.com>”
            <table border="0" width="100%">

            <xsl:apply-templates/>

            </table>
        </form>
        </body>
        </html>
    </xsl:template>

    <xsl:template match="pessoa">
        <tr>
            <td bgcolor="lightgrey" colspan="3">
                        <center><b>Dados Pessoais</b></center>
                    </td>
        </tr>
        <tr>
            <td width="75">Nome:</td>
            <td colspan="2">
                        <input type="text" name="nome">
                          <xsl:attribute name="value">
                             <xsl:value-of select="nome"/>
                          </xsl:attribute></input>
                    </td>
        </tr>
        <tr>
            <td width="75">RG:</td>
            <td width="75">
                        <input type="text" name="rg">
                          <xsl:attribute name="value">
                            <xsl:value-of select="rg"/>
                          </xsl:attribute></input>
                    </td>
        </tr>
    </xsl:template>
```

Figura 3.3: Documento XSLT para geração de formulário HTML

Uma vez que o formulário gerado tenha sido preenchido / atualizado pelo usuário, um processo é disparado para que o modelo seja atualizado com os dados informados. O esquema pode ser visto na Figura 3.4.

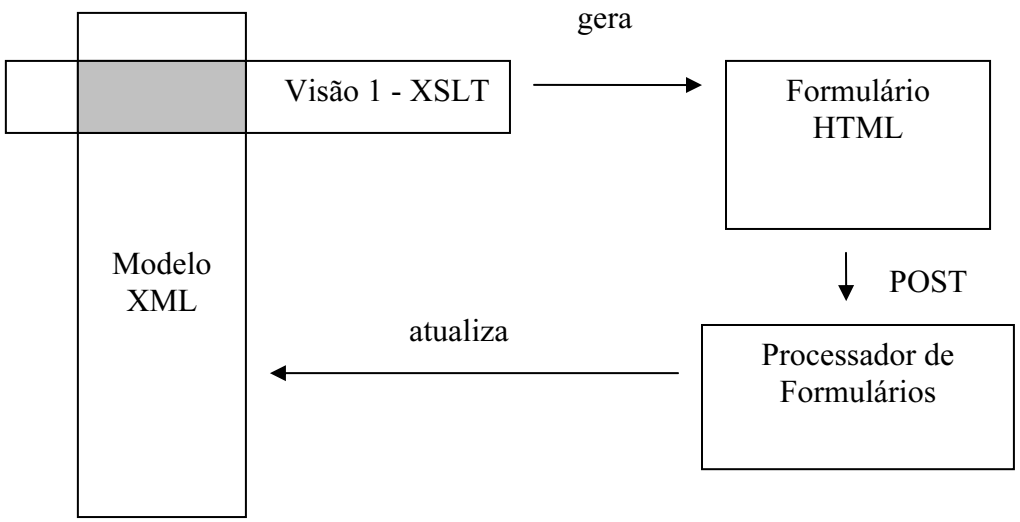


Figura 3.4: Esquema da Solução XML/XSLT

Vamos abordar cada um dos problemas e limitações encontrados em formulários HTML tradicionais e verificar como se comportam nesta arquitetura.

a) Mescla de diferentes componentes: modelo, visão e controlador

Observa-se claramente a separação do modelo dos demais componentes da aplicação. O modelo é definido por um documento XML que possui a estrutura de dados que contém as informações do formulário.

Nota-se que a definição dos componentes de visão e controlador, entretanto, mesclam-se na estrutura de transformação XSLT. No documento XSLT são especificados tanto a forma como o formulário (ou etapa de formulário) será exibida quanto algumas regras comportamentais do mesmo: para onde os dados serão postados, pequenas regras de validação, etc.

b) Dependência de dispositivos

A separação do modelo garante sua independência de dispositivo de hardware / software. Entretanto, a transformação XSLT é realizada com o objetivo de atingir uma determinada plataforma HTML, em um determinado dispositivo. O layout gerado, quanto mais complexo for o formulário, mais dependente de dispositivo tende a ser.

c) Arquitetura de duas camadas

A possibilidade de aplicar transformações diferentes sobre o mesmo modelo é um diferencial em relação à formulários HTML tradicionais. Pode-se facilmente implementar etapas de preenchimento que quebrem o paradigma de duas camadas, facilitando o desenvolvimento de sistemas de workflow. Cada etapa apresenta apenas a visão necessária sobre o modelo, para diferentes personagens da aplicação

d) Dependência de Linguagens de Script

Da mesma forma que em formulários HTML, a implementação do componente controlador é tipicamente realizada através de scripts. O componente visão é gerado a partir de transformações XSLT, mas a visão em si é implementada em formulários HTML tradicionais.

e) Repetitividade e Retrabalho

A padronização do modelo em um único documento XML, não interessando o número de etapas de preenchimento, permite a utilização de serviços comuns no lado servidor para persistência, mapeamento de dados semi-estruturados para estruturados, validações de segurança, entre outras.

Existe ainda retrabalho e repetitividade nas tarefas de implementação do componente controlador, que é implementada através de transformações XSLT. Tipicamente, pequenas regras e validações devem ser implementadas no XSLT, o que acaba se repetindo por diversas aplicações distintas.

f) Dificuldade de Inicialização de Formulários

A separação do modelo é suficiente para resolver o problema da dificuldade de inicialização de formulários. O modelo não está mais embutido dentro do documento do formulário, mas sim separado em um documento XML. A inicialização de uma outra instância de formulário requer apenas a troca do documento XML modelo que se deseja visualizar.

g) Formato de Codificação

O modelo implementado em documentos XML possui uma representação bastante robusta para suportar diferentes necessidades do negócio. Não existe necessidade de uso de artifícios para a nomenclatura dos elementos, como havia em formulários HTML tradicionais.

3.3.2 Técnicas e Ferramentas de Desenvolvimento Baseadas em XML Schema

O XML Schema é um padrão W3C para a definição classes de documentos XML (W3C, 2003). Define desde a estrutura básica do documento até regras de validação, domínio e tipo de dados dos elementos.

O uso de um esquema XML para a definição do componente modelo pode também ser utilizado para a definição do componente controlador da aplicação, com algumas limitações como veremos adiante.

Um exemplo de implementação desta arquitetura em código aberto é o software Schemox (INFOZONE, 2003). A partir das informações disponíveis no esquema XML, formulários genéricos são construídos em formato HTML ou swing. A geração de formulários é realizada por uma classe Java, responsável pela interpretação do esquema XML e geração do formulário HTML ou swing. A estrutura pode ser vista na Figura 3.5.

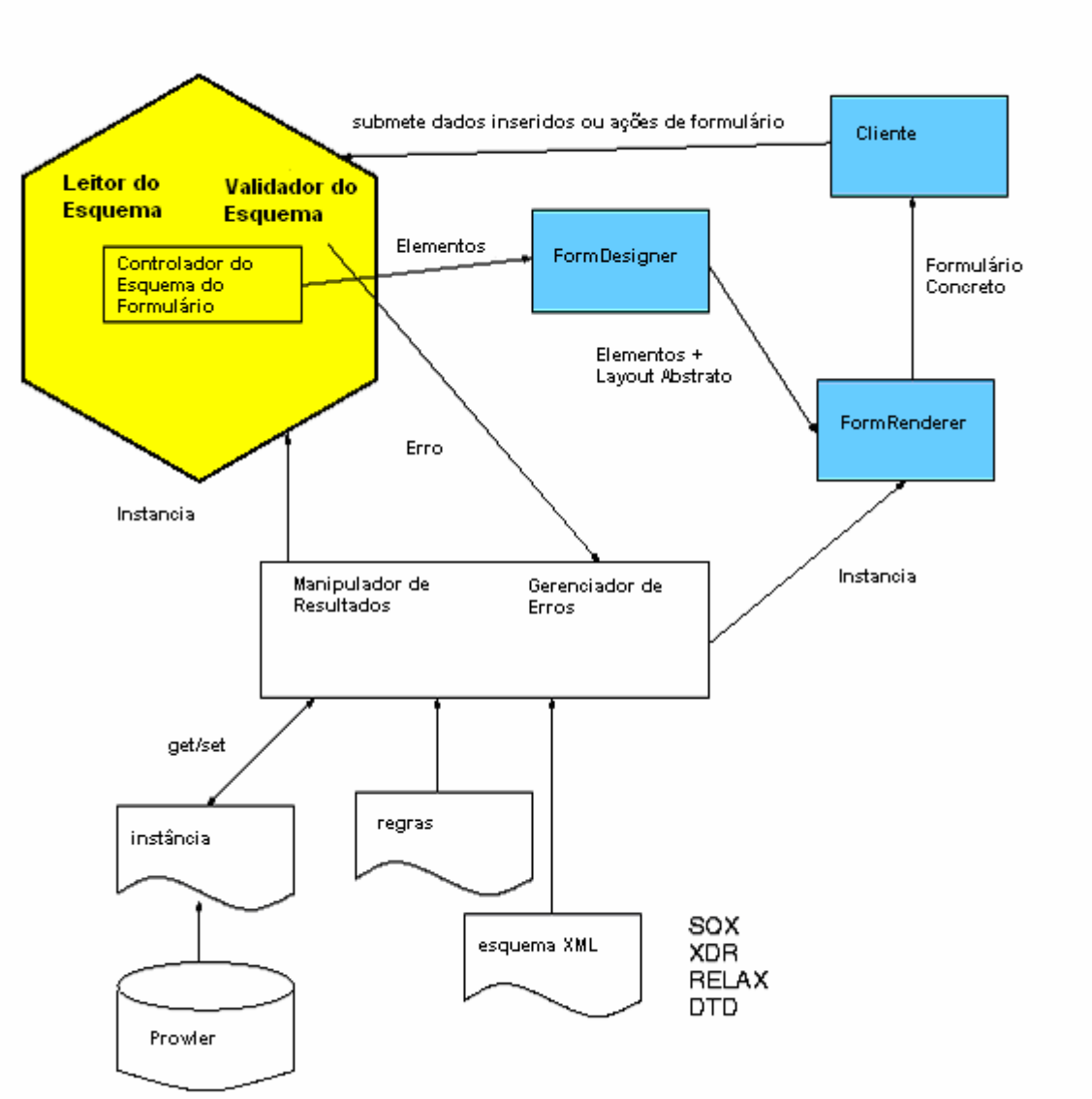


Figura 3.5: Estrutura da implementação SchemoX (INFOZONE, 2003).

Nota-se que a geração do componente visão poderia ser, teoricamente, realizada por uma transformação XSLT sobre o esquema XML definido. Entretanto, esta transformação poderia ser bastante complexa, a ponto de não justificar o esforço dispendido.

Vamos agora analisar como os problemas e limitações de formulários HTML tradicionais se comportam nesta arquitetura.

a) Mescla de diferentes componentes: modelo, visão e controlador

O modelo nesta arquitetura é implementado em um documento XML. O componente controlador é implementado no esquema XML, em um documento separado. A visão é também definida pelo esquema XML, gerando interfaces com o usuário em HTML ou qualquer outra linguagem de apresentação.

Percebe-se uma limitação nas funcionalidades que podem ser implementadas no componente controlador, já que o XML Schema não foi desenvolvido para este fim – mas para a definição de classes de documentos XML. O componente visão fica também limitado, já que também é definido pelo esquema XML.

b) Dependência de dispositivos

A separação do componente modelo também garante sua independência de dispositivo de hardware / software. O XML schema também é independente de dispositivo. A dependência de dispositivo fica totalmente localizada no framework que transforma o esquema XML em componente visão, e que mapeia os dados informados pelo usuário para o componente modelo da aplicação.

c) Arquitetura de duas camadas

O esquema XML não dispõe de recursos para quebrar um formulário em etapas ou rotear formulário. Preserva-se a arquitetura de duas camadas nesta solução, ou seja, o formulário é inteiramente preenchido em uma única visão, processado e armazenado no modelo de uma única vez.

d) Dependência de Linguagens de Script

As regras de validação, tipos de dados e domínios são definidos no esquema XML, independente de linguagem de script. O mecanismo de geração de formulários é responsável pela geração das estruturas necessárias de validação, sejam scripts ou não. A solução, portanto, não é afetada pela dependência de linguagens de scripts, a responsabilidade é do framework que estamos utilizando.

e) Repetitividade e Retrabalho

Da mesma forma que em soluções que utilizam XSL, a padronização do modelo em uma instância de documento XML permite a utilização de serviços comuns do lado servidor para persistência, mapeamento de dados semi-estruturados para estruturados, validações de segurança, entre outras.

Além disso, o componente controlador também é beneficiado pelo uso de recursos comuns de validação de domínios e tipos de dados.

f) Dificuldade de Inicialização de Formulários

A separação do modelo é suficiente para resolver o problema da dificuldade de inicialização de formulários, assim como na solução que utiliza XSL puro. O fato do modelo não estar embutido dentro dos demais componentes requer apenas a troca da instância de documento XML para que uma outra instância de formulário seja aberta.

g) Formato de Codificação

O modelo implementado em documentos XML, assim como na solução XSL, é robusta o suficiente para suportar diferentes necessidades do negócio.

3.3.3 Técnicas de Desenvolvimento Baseadas em XML e DOM

A solução baseada em XML e DOM é similar à solução com XML schema: utiliza uma instância de documento XML para o componente modelo. Entretanto, ao invés de usar um esquema XML para a definição dos componentes controlador e visão, utiliza-se uma outra instância de documento XML para esta tarefa. Um módulo construtor de formulários, tipicamente implementado usando DOM (W3C, 2003), é responsável por construir formulários HTML baseado na especificação XML e no componente modelo.

Como visto na seção anterior, a solução com XML schema introduz muitas limitações quanto à recursos nos componentes controlador e visão. O XML schema não foi feito para isso, mas sim para a definição de classes de documentos XML (W3C, 2003). A proposta desta solução é eliminar as limitações impostas pelo XML schema quando usado desta forma.

A arquitetura desta solução pode ser vista na Figura 3.6.

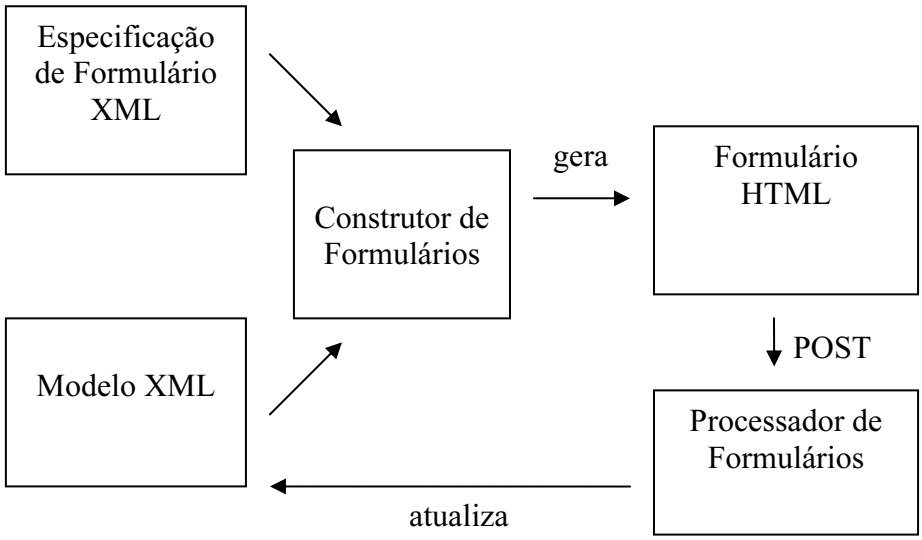


Figura 3.6: Arquitetura da solução

A solução proposta por Fitch (2002) é uma implementação nesta arquitetura, sendo o construtor de formulários implementado inteiramente no navegador Internet Explorer. A especificação de formulário é implementada por um documento XML schema estendido com informações de apresentação e validação.

Vamos analisar a arquitetura proposta contra os problema e limitações identificados em formulários HTML tradicionais.

a) Mescla de diferentes componentes: modelo, visão e controlador

Assim como na arquitetura que utiliza XML schema, o modelo é implementado em um documento XML separado. O componente controlador é implementado em um documento XML de especificação de formulário, que também define a forma como será apresentado o formulário. O uso de uma especificação de formulário não rígida como o XML schema permite a definição de componentes visão bem mais complexos.

b) Dependência de dispositivos

Assim como na implementação com o XML schema, o modelo é totalmente independente de implementação em diferentes arquiteturas de hardware / software.

A especificação do formulário é, porém, direcionada para uma determinada implementação, já que contém detalhes de como deve ser apresentado o formulário para o usuário.

c) Arquitetura de duas camadas

A possibilidade de criar a especificação do formulário na estrutura desejada possibilita a quebra do preenchimento do formulário em etapas, com visões específicas sobre o modelo. A criatividade do autor fica aberta para quebrar o paradigma de duas camadas existente em formulários HTML tradicionais.

d) Dependência de Linguagens de Script

Da mesma forma que na solução por esquemas XML, a implementação da lógica de controle é definida no módulo construtor de formulários. O construtor de formulários é uma implementação genérica comum a todas as aplicações. Possivelmente usará recursos de linguagens de scripts para executar validações e garantir algumas regras no preenchimento do formulário.

A especificação do formulário (controlador) e o modelo, entretanto, são totalmente independentes de linguagens de script.

e) Repetitividade e Retrabalho

Os benefícios obtidos no uso do XML schema é também válido para a solução XML / DOM. Diversos serviços comuns podem ser definidos reduzindo o retrabalho na definição de novas aplicações. Pode-se especificar não somente serviços de validação de domínios e tipos de dados, mas qualquer operação desejada. Um exemplo disto é a definição de uma interface comum com outras aplicações / banco de dados. Basta definir como o formulário deve se comportar para buscar dados de outras aplicações na especificação do formulário, e um serviço específico utiliza essas informações para executar esta operação.

f) Dificuldade de Inicialização de Formulários

Assim como no esquema XML, a separação do modelo é suficiente para resolver o problema da dificuldade de inicialização de formulários. A simples troca da instância de documento XML que contém o modelo é suficiente para que uma nova instância de formulário seja aberta.

g) Formato de Codificação

O modelo sendo implementado em instâncias de documentos XML possui uma estrutura de dados bastante rica, suficiente para suportar diferentes necessidades do negócio.

4 O PADRÃO XFORMS 1.0

O padrão XForms (DUBINKO et al., 2002) é um reconhecimento do W3C da importância que os formulários possuem na Internet, como meio principal para a elaboração de aplicações web interativas.

Neste capítulo é introduzido o padrão XForms, apresentar as vantagens do XForms em relação às demais técnicas de desenvolvimento de formulários, e, por último, apresentar algumas implementações já existentes do padrão XForms. No final do capítulo é mostrado o racional de escolha da implementação que será utilizada durante o desenvolvimento da aplicação piloto.

4.1 Histórico do XForms

A primeira especificação rascunho do padrão XForms foi apresentada em abril de 2000 (W3C, 2000). O foco desta publicação era essencialmente a representação de modelos de dados para o padrão XForms, baseados no trabalho sobre esquemas XML, que estava sendo desenvolvido na mesma época. Apesar de similares, alguns tipos de dados que foram definidos para o modelo de dados no padrão XForms eram diferentes dos definidos pelo esquema XML, devido a diferentes cenários e público alvo (W3C, 2000).

Em pouco tempo o W3C percebeu o problema que estava sendo criado com as diferenças especificadas nos dois trabalhos, e ainda no ano de 2000 reconheceu a necessidade de utilizar especificações XML schema puras para o modelo de dados.

Os trabalhos continuaram, com a definição dos módulos de interface com o usuário e a utilização de tipos de dados simples dos esquemas XML para o modelo de dados. Em 2001 o modelo começou a adquirir maturidade, com autores proporcionando feed-backs para o W3C. Finalmente, em novembro de 2002, a recomendação candidata XForms 1.0 foi finalizada com uma chamada para implementações.

Em abril de 2003 a recomendação candidata estava preparada para o estágio de recomendação proposta.

4.2 Objetivos do Padrão XForms

Nesta seção serão detalhados os objetivos chave do padrão XForms, como descritos na especificação.

4.2.1 Suporte a handhelds, televisão, e desktops, bem como impressoras e scanners

O padrão XForms propõe a especificação de formulários independentemente do dispositivo que será utilizado para visualizá-lo. O mesmo formulário pode ser exibido em qualquer dispositivo onde a *engine* XForms tenha sido implementada.

Exemplos de implementação em diferentes dispositivos incluem handhelds, desktops, televisões, telefones e sistemas de áudio. Os elementos do formulário são implementados de forma diferente em cada dispositivo, de acordo com a melhor forma possível de apresentação para o usuário final.

Até o presente momento não foram identificadas implementações do padrão XForms em plataformas não PC, portanto a aplicação piloto não avaliará este objetivo chave.

4.2.2 Rica interface com o usuário, que atinja as necessidades de negócio, consumidores, e aplicações de controle de dispositivos

A interface com o usuário deverá atender plenamente os requisitos da aplicação. O conjunto de controles disponibilizados pelo modelo deve atender essa necessidade, desde a apresentação até a semântica de cada controle.

Para a realização deste trabalho, a interface com o usuário será verificada através da escolha de uma aplicação piloto complexa. Os seguintes aspectos do padrão XForms serão verificados:

- **Tipos de Dados vinculados a controles**. O padrão XForms possibilita a definição de tipos de dados no esquema do modelo, o que pode se traduzir em diferentes visualizações para um mesmo controle XForms, dependendo do tipo de dado escolhido. A aplicação piloto deverá fazer uso deste recurso.
- **Elementos de Repetição**. O padrão XForms permite a especificação de elementos de repetição, que possibilitam a entrada múltipla de informações na aplicação. Um exemplo de tal uso é a existência de múltiplos itens em um pedido. A especificação do padrão XForms para a repetição de elementos será verificada na aplicação piloto
- **Controles de Seleção Unitária**: As caixas de seleção de uma entre várias opções disponíveis é um dos elementos mais importantes nas aplicações com formulários. A aplicação piloto fará uso dos controles de seleção unitária
- **Controles relacionados**: O relacionamento entre campos é um dos aspectos mais complexos em aplicações baseadas em formulários. Um exemplo clássico de relacionamento entre campos são caixas de seleção de País, Estado e Município. Uma vez definido o País, a caixa de seleção de Estados deverá ser preenchida. Por sua vez, quando o Estado for selecionado, a caixa de seleção de municípios deverá ser preenchida. A quantidade de opções na caixa de seleção de hierarquia mais baixa é exponencial, e a aplicação precisa tratar estes casos adequadamente. A aplicação piloto fará uso de controles relacionados.

4.2.3 Dados, lógica e apresentação desacoplados

O padrão XForms prevê o desacoplamento dos dados capturados, da lógica e da apresentação do formulário. O resultado obtido deve simplificar a compreensão e a manutenção da aplicação. A aplicação piloto verificará os resultados obtidos com o desacoplamento dos três aspectos.

4.2.4 Internacionalização melhorada

O padrão XForms deve facilitar a internacionalização de formulários, possibilitando a entrada de dados em conjuntos de caracteres não ocidentais, bem como facilitar a tradução da aplicação para diversos idiomas. A aplicação piloto verificará a implementação da internacionalização no padrão XForms

4.2.5 Suporte a dados de formulários estruturados

A integração com aplicações legadas, que utilizem fontes de dados estruturadas, é um requisito essencial para o padrão XForms. A aplicação piloto deverá integrar-se com outras fontes de dados, inclusive estruturadas.

4.2.6 Lógica de formulários avançada

O padrão XForms deve atender as necessidades dos desenvolvedores, disponibilizando um padrão que possibilite a implementação de apresentações complexas. A aplicação piloto, através da interface rica com o usuário, fará uso de estruturas de apresentação avançadas, como campos relacionados.

4.2.7 Múltiplos formulários por página, e páginas por formulário

Um dos problemas clássicos em formulários web tradicionais são os formulários extensos. A quebra de um formulário web em múltiplas etapas, ou páginas de preenchimento, normalmente é um requisito de complexidade elevada. O padrão XForms propõe minimizar o esforço gasto com esta tarefa. A aplicação piloto utilizará múltiplas páginas por formulário, com a intenção de verificar a eficácia introduzida pelo padrão.

O padrão XForms deve ainda possibilitar múltiplos formulários em uma mesma página, independentes ou não

4.2.8 Suporte a operações de suspender / continuar

O padrão XForms propõe a solução de um problema clássico em formulários web tradicionais: a possibilidade de continuar o preenchimento de um formulário a partir de um determinado ponto de interrupção. A aplicação piloto tentará implementar a operação de suspender / resumir preenchimento de formulários.

4.2.9 Fácil integração com outros conjuntos de elementos XML

Sendo uma especificação baseada em XML, o padrão XForms deve integrar-se facilmente a outros espaços de nomes XML, bem como demais dialetos. Transformações XSLT serão utilizadas na aplicação piloto, bem como serão utilizados outros espaços de nome necessários para o funcionamento da mesma.

4.3 Arquitetura do Padrão XForms

O padrão XForms separa o que o formulário faz, como o formulário se apresenta, e de onde os dados são manipulados. Com essa separação, equipes distintas podem colaborar com maior produtividade no desenvolvimento de uma solução completa envolvendo formulários.

Na Figura 4.1 pode-se visualizar um exemplo da separação do modelo XForms da forma de apresentação. Um mesmo modelo pode ser apresentado de diversas formas, atendendo diversos dispositivos.

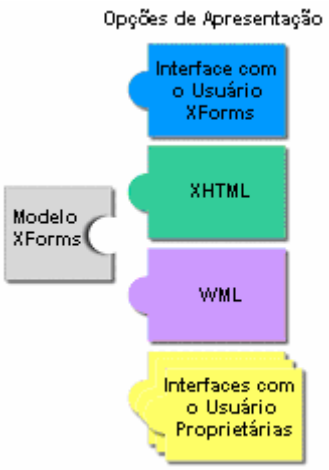


Figura 4.1: Separação do Modelo XForms da Apresentação (MICHEL, 2003)

Os dados coletados no padrão XForms são armazenados em instâncias de documentos XML, que seguem a uma especificação inicialmente definida no modelo. Através de um protocolo de submissão, o XForms é capaz de enviar e receber dados, como pode-se ver na Figura 4.2.

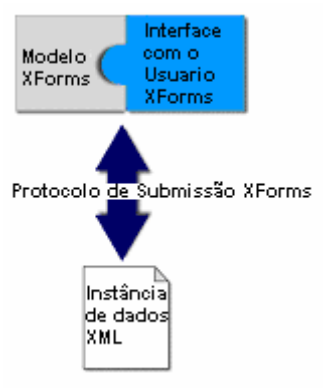


Figura 4.2: Protocolo de Submissão XForms (MICHEL, 2003)

O padrão XForms compreende dois dialetos em XML. O primeiro deles define o modelo do formulário, a estrutura de dados que manterá as informações preenchidas. O segundo descreve como o formulário será apresentado ao usuário (ORBEON, 2003).

Na Figura 4.3 podemos ver um exemplo de interface com o usuário para o preenchimento de informações relativas a cartão de crédito, em uma loja virtual:

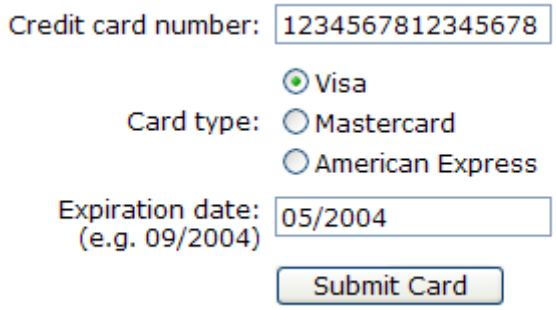


Figura 4.3: Exemplo de Interface XForms com o Usuário (ORBEON, 2003)

O modelo XML define o documento XML usado para capturar o que foi digitado pelo usuário (chamado de instância), como o formulário é submetido (qual é a URI e o método usado – GET ou POST), e como os dados informados podem ser validados (ORBEON, 2003).

O exemplo da Figura 4.4 mostra como um documento de modelo XForms seria definido para o formulário de cartão de crédito acima.

```
- <xforms:model schema="credit-card-info.rng"
xmlns:xforms="http://www.w3.org/2002/01/xforms">
    - <xforms:instance>
        - <credit-card-info>
            <number/>
            <type>visa</type>
            <expiration-date/>
          </credit-card-info>
      </xforms:instance>
      <xforms:submission action="http://www.example.com/submit-card" method="post"/>
  </xforms:model>
```

Figura 4.4: Exemplo de Modelo XForms (ORBEON, 2003)

Neste exemplo, algumas definições básicas são realizadas (ORBEON, 2003):

- A estrutura geral da instância: um elemento credit-card-info contendo outros três elementos: number, type e expiration-date
- O esquema a ser usado para validar o formulário submetido pelo usuário: credit-card-info.rng
- A URI para onde o formulário será submetido: (http://www.example.com/submit-card), e o método usado (POST)

Além do modelo, é necessário definir quais controles serão utilizados, por exemplo, campos textos ou botões de rádio, assim como a forma que esses controles serão ligados à instância (por exemplo, ligar o campo “Credit card number” ao elemento “number” na instância.

No exemplo da Figura 4.5, podemos ver como o XForms poderia ser utilizado para criar os controles apropriados para o exemplo do cartão de crédito (ORBEON, 2003).

```
- <xforms:group xmlns:xforms="http://www.w3.org/2002/01/xforms">
      <xforms:input ref="credit-card-info"/>
    - <xforms:submit>
        <xforms:input ref="number"/>
      - <xforms:select1 ref="type" appearance="full">
          - <xforms:choices>
              - <xforms:item>
                  <xforms:caption>Visa</xforms:caption>
                  <xforms:value>visa</xforms:value>
                </xforms:item>
              - <xforms:item>
                  <xforms:caption>Mastercard</xforms:caption>
                  <xforms:value>mastercard</xforms:value>
                </xforms:item>
              - <xforms:item>
                  <xforms:caption>American Express</xforms:caption>
                  <xforms:value>amex</xforms:value>
                </xforms:item>
            </xforms:choices>
        </xforms:select1>
        <xforms:input ref="expiration-date"/>
    </xforms:submit>
</xforms:group>
```

Figura 4.5: Exemplo de Definição de Interface XForms (ORBEON, 2003)

4.4 Vantagens do Padrão XForms em Relação a Outras Implementações

Para melhor descrever as vantagens do Padrão XForms, vamos analisá-lo quanto aos problemas e limitações encontrados no desenvolvimento de aplicações que utilizem formulários HTML.

4.4.1 Mescla de diferentes componentes: modelo, visão e controlador

O Padrão XForms possibilita a separação dos componentes MVC por completo. O modelo XForms define, entre outras coisas, a estrutura de dados em um documento XML que será criado, chamado de instância. Um exemplo de modelo XForms pode ser visto na Figura 4.4. O resultado da instância XML gerada pode ser vista na Figura 4.6.

```
- <credit-card-info>
        <number>1234567812345678</number>
        <type>visa</type>
        <expiration-date>05/2002</expiration-date>
  </credit-card-info>
```

Figura 4.6: Exemplo de Instância gerada pelo XForms (ORBEON, 2003)

A visão da estrutura MVC é definida pelo controle XForms, que especificará os tipos de controles que serão usados, bem como a ligação com o componente modelo. Na Figura 4.5 pode-se ver um exemplo de controle XForms que define a forma como se apresentará o formulário.

O controlador da estrutura MVC é implementado pela *engine* do XForms, que manipula a mescla dos componentes visão e modelo para a apresentação de formulários para o usuário, assim como trata requisições de atualização feitas pelo usuário. Nota-se que o mapeamento da visão para o modelo está definida no documento de controle XForms.

4.4.2 Dependência de Dispositivos

Como pode ser visto na Figura 4.1, o padrão XForms prevê a separação do modelo XForms da apresentação. Isso significa que a especificação do modelo, assim como a especificação dos controles XForms são independentes de plataforma de hardware / software. Cabe a implementação das opções de apresentação da *engine* XForms mapear as definições do modelo e controle a um determinado dispositivo.

4.4.3 Arquitetura de Duas Camadas

O padrão XForms possibilita uma visão mais flexível do que a arquitetura de duas camadas, possibilitando que a instância com os dados do formulário seja roteada para várias estações, conforme necessário (DUBINKO, 2003). Em cada ponto do workflow, os dados são carregados em um formulário, que provê uma visão para edição de todo ou

parte do documento, sendo ao final submetido novamente. Estes passos podem ser repetidos tantas vezes quantos forem necessários, com qualquer número de participantes.

4.4.4 Dependência de Linguagens de Scripts

O padrão XForms ajuda a reduzir a necessidade de scripts de diversas formas (DUBINKO, 2003). A primeira delas é definir dentro do framework cálculos e validações simples baseadas em XPath. O autor não precisa escrever rotinas em linguagens de script, basta especificá-las usando XPath. Um exemplo pode ser visto na Figura 4.7, onde pode-se calcular para cada linha de um pedido o preço total de cada item.

```
<!-- Para cada linha do pedido, calcula o preço total do item -->
<xforms:bind nodeset="itens/item/ext" calculate="../preco * ../qtde"/>
```

Figura 4.7: Exemplo de uso de XPath em XForms

Além disso, um melhor retorno para o usuário sobre as atividades do formulário é provido automaticamente pela *engine* do XForms. A *engine* XForms pode automaticamente avisar que determinadas ações estão sendo realizadas (por exemplo, o formulário está sendo submetido ao servidor). Tabelas repetitivas, seções opcionais e um conjunto de ações comuns está disponível para serem usados pelo autor do formulário, sem necessidade de usar linguagens de script.

4.4.5 Repetitividade e Retrabalho

A *engine* do XForms implementa diversos serviços comuns a todas as aplicações que utilizam formulários XForms. Como exemplo, a validação de segurança da informação contra a entrada de dados maliciosos pode ser implementada diretamente na *engine*, sem a necessidade do autor se preocupar com isto.

O padrão XForms procura deixar para o autor o design de código referente à aplicação, não devendo se preocupar com tarefa comuns realizadas para qualquer tipo de aplicação.

4.4.6 Dificuldade de Inicialização de Formulários

No padrão XForms, os dados do formulário são providenciados por um documento XML incial (a instância) (DUBINKO, 2003). Já que o padrão XForms é flexível o suficiente para manipular diretamente com a maioria dos formatos XML, utilizar dados iniciais em um formulário é tão simples quanto alterar um atributo de fonte que apontará para uma determinada instância de documento XML

4.4.7 Formato de Codificação

Como visto na Figura 4.6, o componente modelo é implementado como um documento XML. O XML é uma fundação melhor para a maioria dos documentos de negócio do que pares de nome / valor (DUBINKO, 2003). Conseqüentemente, o autor terá uma estrutura poderosa para representar suas necessidades dentro de formulários.

4.5 Detalhamento do Padrão XForms

Nesta seção será examinado brevemente o padrão XForms, com uma descrição dos elementos principais que o compõe. Para uma descrição detalhada, sugere-se a descrição do padrão XForms (MICHEL, 2003).

Como visto na Figura 4.2, o padrão XForms separa uma seção para o modelo, que descreve o que o formulário faz, e outra seção para a apresentação, que descreve como o formulário será apresentado. Pode-se observar ainda a existência de um protocolo de submissão e, por último, a instância XML com os dados capturados.

4.5.1 O Modelo XForms

O modelo XForms define o comportamento do formulário que será utilizado para gerar / manipular instâncias de documentos XML com os dados capturados. Define ainda regras de negócio que estão implementadas no formulário.

O modelo XForms é similar ao modelo na arquitetura modelo – visão – controlador. De fato, o padrão XForms foi projetado tendo-se em mente a arquitetura MVC.

Dentro do modelo deve-se especificar a instância de dados que será manipulada, para onde os dados serão submetidos e amarrações entre elementos.

Na Figura 4.8, podemos visualizar um exemplo de modelo XForms.

```
<xforms:model id="t1">
    <xforms:submission action="file://tmp/itensVenda.xml" method="put" id="submit"/>
    <xforms:instance>
        <itens>
            <item>
                <valor>2</valor>
                <preco>2.30</preco>
                <total>0</total>
            </item>
            <item>
                <valor>3</valor>
                <preco>1.20</preco>
                <total>0</total>
            </item>
        </itens>
    </xforms:instance>
    <xforms:bind nodeset="item/total" calculate="../valor * ../preco" />
</xforms:model>
```

Figura 4.8: Exemplo de Modelo XForms

A definição do modelo é feita pelo elemento **<xforms:model>**. Observa-se que o mesmo possui três elementos distintos: <**xforms:submission**>, <**xforms:instance**> e <**xforms:bind**>.

O elemento <**xforms:submission**> define como, para onde, e o quê submeter do formulário (MICHEL, 2003). O atributo *action* define para onde será submetido o formulário, da mesma forma que em atributos *action* de formulários HTML. O atributo *method*, também espelhando-se em formulários HTML, define como os dados serão submetidos. Através do atributo *ref*, não exemplificado na Figura 4.8, poderia-se escolher determinadas porções da instância de dados para submeter, ao invés da instância inteira.

O elemento <**xforms:instance**>, por sua vez, define a instância de dados XML inicial que será utilizada para popular o formulário. No exemplo da Figura 4.8, a instância estava diretamente definida dentro do documento principal, mas poderia-se especificar apenas uma referência (URI) ao documento que contém a instância. Essa abordagem pode ser vista na Figura 4.9, estando a instância separada representada na Figura 4.10.

```
<xforms:model id="t1">
    <xforms:submission action="file://tmp/itensVenda.xml" method="put" id="submit"/>
    <xforms:instance src="itensVenda.xml"/>
    <xforms:bind nodeset="item/total" calculate="../valor * ../preco" />
</xforms:model>
```

Figura 4.9: Exemplo de Modelo XForms Simplificado

```
<?xml version="1.0" encoding="UTF-8"?>
<itens>
    <item>
        <valor>2</valor>
        <preco>2.30</preco>
        <total>0</total>
    </item>
    <item>
        <valor>3</valor>
        <preco>1.20</preco>
        <total>0</total>
    </item>
</itens>
```

Figura 4.10: Instância de dados XML Separada do Modelo

O elemento <**xforms:bind**> permite amarrar um nodo selecionado dos dados da instância e atribuir algum valor a ele. No exemplo da Figura 4.8, um cálculo do preço total de cada item é realizado e armazenado no elemento **total** correspondente.

4.5.2 A Interface com o Usuário no padrão XForms

A interface com o usuário no padrão XForms é o componente visão na arquitetura MVC. Através dela é definida a apresentação do formulário, embora independente de

dispositivo. Os elementos de apresentação são definidos em alto nível, de forma que cada implementação possa realizar o mapeamento para cada dispositivo específico.

Uma descrição detalhada dos controles de formulários disponíveis no padrão XForms está além do escopo deste trabalho. A definição do padrão (MICHEL, 2003) apresenta todos os elementos disponíveis. Vamos nesta seção apresentar os conceitos envolvidos com a interface com o usuário no padrão XForms.

O conceito fundamental na interface com o usuário no padrão XForms é o de "controle de formulário". Os controles de formulário são janelas para os dados do formulário mantidos no Modelo XForms. Os controles de formulário XForms podem ser vistos na Tabela 4.1.

Tabela 4.1: Controles de Formulário no padrão XForms

Controle de Formulário	Descrição Sucinta
input	Similar ao controle input dos formulários HTML, permite a entrada de quaisquer caracteres. Tipos de dados podem ser atrelados de forma a melhorar a apresentação (por exemplo, calendários para campos *input* com tipo de dados data).
secret	Praticamente idêntico ao campo *input* do tipo *password* no HTML, permite a entrada de dados sem que os mesmos apareçam na tela.
textarea	Muito parecido com o *textarea* no HTML, permitindo a entrada de dados em múltiplas linhas
output	Não permite a entrada de dados pelo usuário, apenas exibe dados para o usuário, seguindo as mesmas regras de apresentação dos demais elementos de formulários XForms
upload	Permite não apenas a seleção de arquivos para submissão ao servidor, como também a integração com equipamentos de digitalização, leitores de cartão, microfones, câmeras, entre outros
range	Não disponível em formulários HTML, apresenta uma forma intuitiva de entrada de dados com valores limites inferiores e superiores definidos.
trigger	Similar ao elemento *button* dos formulários HTML, e dispara alguma ação no formulário. Não restringe-se apenas a botões
submit	Inicia a submissão dos dados da instância, total ou parcialmente. É uma especialização do controle *trigger*.
select	Permite ao usuário selecionar múltiplos itens de um conjunto de opções. É equivalente em formulários HTML a uma lista que permite a seleção de múltiplos itens, ou a um conjunto de *checkboxes* referentes a um tema específico
select1	Permite a seleção de apenas um elemento em uma lista de múltiplas escolhas. Equivalente a uma lista em formulários HTML, com única escolha, ou a um conjunto de botões de rádio.
choices	Usado dentro de controles de seleção para agrupar um conjunto de opções relacionadas. Provê a mesma funcionalidade do elemento *optgroup* em formulários HTML
item	Especifica o valor de armazenamento e o *label* para representar um item em uma lista
filename	Usado em conjunto com o controle *upload* para representar um recurso binário disponível
mediatype	Usado em conjunto com o controle *upload*, que representa o tipo de recurso binário selecionado
value	Especifica o valor de armazenamento para ser usado quando um item é selecionado.
label	Coloca uma etiqueta descritiva no controle de formulário a que se refere
help	Provê uma maneira conveniente para atrelar algum texto de ajuda ao controle de formulário
hint	Provê uma maneira conveniente para atrelar alguma dica de preenchimento ao controle de formulário
alert	Define informações ou mensagens de erro relativos ao preenchimento de um determinado controle de formulário.

Os controles de formulários definidos na Tabela 4.1 são tratados como unidades individuais, para o propósito de layout visual (MICHEL, 2003). Eventualmente pode surgir a necessidade de agrupar controles em um formulário. Essa informação poderia ser útil, por exemplo, para pequenos dispositivos. A informação de agrupamento

poderia ser utilizada para determinar o que deve ser apresentado na mesma página, caso seja necessário quebrar o formulário em diversas páginas. O elemento **group** é usado para definir um agrupamento no padrão XForms.

Um agrupamento pode ainda ser usado para definir o escopo de trabalho dentro da estrutura do modelo. Através do atributo **ref** pode-se definir, como conveniência, o caminho relativo XPath que os elementos filhos do grupo farão referência. Um exemplo de agrupamento pode ser visto na Figura 4.11. Neste exemplo, o escopo do modelo usado dentro do grupo é o elemento *enviarPara*. O elemento *input* dentro do grupo estabelece expressões XPath relativas a este elemento.

```
<ordemCompra>
  <enviarPara>
      <Cidade>Porto Alegre</Cidade>
  </enviarPara>
</ordemCompra>

<group ref="enviarPara">
  <input ref="Cidade"></input>
</group>
```

Figura 4.11: Exemplo de Agrupamento no padrão XForms

Para os casos em que se necessite implementar uma apresentação dinâmica, como, por exemplo, mostrar apenas porções do formulário somente se uma determinada ação do usuário for disparada, utiliza-se os elementos **switch** e **case**. Em qualquer momento, somente uma instância **case** será processada no documento final (DUBINKO, 2003). A mudança na apresentação de elementos *case* dentro de um *switch* é realizada por uma ação chamada *toggle*, que toma como parâmetro o ID do elemento *case* que será ativado. Um exemplo de apresentação dinâmica pode ser visto na Figura 4.12.

```
<xforms:switch id="sw">
    <xforms:case id="pageA" selected="true">
        <xforms:trigger id="nextButton">
            <xforms:label>Próximo</xforms:label>
            <xforms:action ev:event="xforms-activate">
                <xforms:toggle case="pageB"/>
            </xforms:action>
        </xforms:trigger>
    </xforms:case>
    <xforms:case id="pageB" selected="false">
        <xforms:trigger id="backButton">
            <xforms:label>Voltar</xforms:label>
            <xforms:action ev:event="xforms-activate">
                <xforms:toggle case="pageA"/>
            </xforms:action>
        </xforms:trigger>
    </xforms:case>
</xforms:switch>
```

Figura 4.12: Exemplo de Apresentação Dinâmica no padrão XForms

O padrão XForms dispõe de mais um elemento, utilizado para repetições de conjuntos de controles de formulário. Esta funcionalidade é bastante útil, por exemplo, quando for necessário modelar múltiplos itens em uma ordem de compra.

Através do elemento **repeat**, estabelece-se um agrupamento que será repetido tantas vezes quanto existirem elementos definidos no atributo **nodeset**. Um exemplo de repetição de conjunto pode ser visto na Figura 4.13.

```
<body>
    <xforms:repeat id="repeat1" model="t1" nodeset="item">
        <xforms:input ref="valor">
            <xforms:label>Valor</xforms:label>
        </xforms:input>
        <xforms:input ref="preco" >
            <xforms:label>Preço</xforms:label>
        </xforms:input>
        <xforms:output ref="total">
            <xforms:label>Total</xforms:label>
        </xforms:output>
        <br />
    </xforms:repeat>
</body>
```

Figura 4.13: Exemplo de Repetição de Itens no XForms

4.5.3 O Modelo de Processamento no padrão XForms

O processamento em formulários no padrão XForms é implementado através da definição de respostas a determinados eventos, utilizando para isso as definições de uma outra recomendação candidata do W3C, o XML Events (MCCARRON et al., 2003). Como resposta a um evento, o desenvolvedor pode especificar as ações a serem tomadas.

4.5.3.1 O XML Events

De acordo com a definição (MCCARRON et al., 2003), um evento é a representação de alguma ocorrência assíncrona. Um exemplo de evento poderia ser clicar em algum botão do formulário.

Cada evento é amarrado a um elemento alvo, que representa o principal ponto aonde a ação está acontecendo (DUBINKO, 2003). No exemplo do clique em um botão, o elemento alvo no padrão XForms será o botão que foi clicado. Cada evento pode ainda ter definido um processamento padrão que será realizado.

O comportamento padrão de um evento no XML Events é que deve ser despachado, passando pela estrutura do documento até o elemento onde o evento ocorreu. Esta fase é chamada de captura. Uma vez atingindo o elemento alvo, o evento volta pela estrutura de árvore até a raiz, o que é conhecido como fase *bubbling*. O evento pode ser tratado em qualquer um dos elementos no caminho pelo qual passa, em qualquer fase, podendo inclusive interromper o seu caminho. Na Figura 4.14 podemos ver um esquema do fluxo de eventos típico em XML Events (MCCARRON et al., 2003).

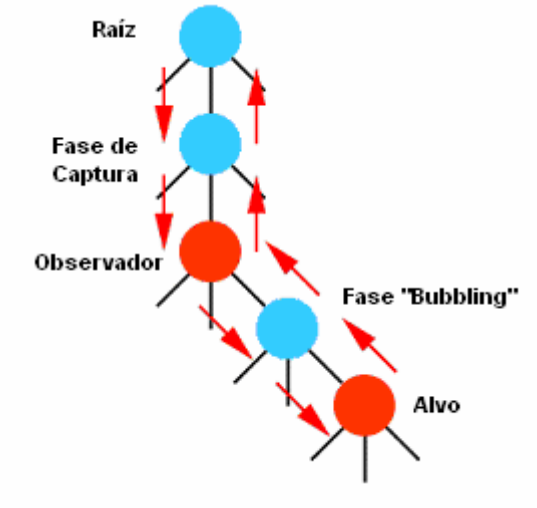


Figura 4.14: Fluxo do Evento no XML Events (MCCARRON et al., 2003).

As definições de quais eventos serão tratados em um determinado elemento podem ser feitas no próprio elemento, através do atributo **event**. Na Figura 4.15 pode-se ver um exemplo de um botão que, ao ser clicado, exibe uma mensagem de teste.

```
<xforms:trigger>
    <xforms:label>Clique Aqui!</xforms:label>
    <xforms:action ev:event="xforms-activate">
        <xforms:message level="modal">Testando</xforms:message>
    </xforms:action>
</xforms:trigger>
```

Figura 4.15: Exemplo de Definição de Resposta a um Evento

A descrição detalhada dos eventos disponíveis no padrão XForms está fora do escopo do presente trabalho. Sugere-se a leitura de Dubinko (2003), que descreve todos os eventos que podem ser observados no padrão XForms.

4.5.3.2 As ações no padrão XForms

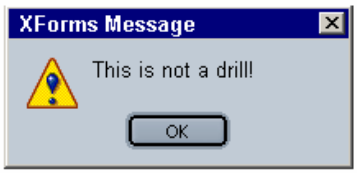
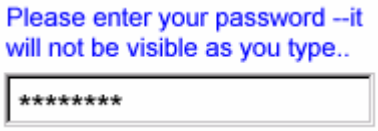
Como resposta a ocorrência de eventos, o XForms disponibiliza um conjunto de ações comumente utilizadas, para evitar a necessidade de escrever scripts de código com as mesmas finalidades. Nesta seção veremos brevemente as principais ações disponíveis.

Ação message

Esta ação envia uma mensagem para o usuário, com diversos níveis de intrusão. (DUBINKO, 2003). Para o nível de intrusão mais alto, que obriga o usuário a fechar a janela de mensagem antes de continuar a trabalhar no formulário (janela *modal*), é equivalente a função javascript *alert*. Existem ainda níveis de intrusão menores, como

janelas não modais (*modeless*) e pequenas mensagens do tipo dica (ephemeral). Na Tabela 4.2 podemos ver os diferentes resultados para os diferentes níveis de intrusão que se pode especificar em uma ação **message**.

Tabela 4.2: Exemplos de Ação Message

Tipo	Código	Exemplo
Modal	<xforms:message level="modal"> This is not a drill! </xforms:message>	
Modeless	<xforms:message level="modeless"> Please enter your password --it will not be visible as you type </xforms:message>	
Ephemeral	<xforms:message level="ephemeral"> Have you forgotten your password? Simply call 1-900-555-1212 and have A major credit card handy. </xforms:message>	

O conteúdo das mensagens pode ser especificado diretamente no elemento, como nos exemplos acima, ou ainda com referências aos dados da instância, ou então com referências a arquivos externos.

Ação setvalue

Uma ação bastante comum em formulários é atribuir valores a campos, como resultado de algum cálculo XPath, por exemplo, ou copiar dados de algum outro elemento do formulário. Essas ações podem ser atribuídas a ocorrência de determinados eventos. Na Tabela 4.3 pode-se ver exemplos de uso da ação **setvalue**.

Tabela 4.3: Exemplos de Ação Setvalue

Descrição	Exemplo
Setar o valor de um elemento com base em uma expressão XPath	<xforms:setvalue bind="put-here" value="a/b/c"/>
Setar o valor de um elemento para um valor literal	<xforms:setvalue bind="put-here">Teste</xforms:setvalue>
Setar o valor de um elemento como um outro na inicialização do formulário	<xforms:setvalue bind="put-here" value="a/b/c" ev:event="xforms-ready"/>

Ação setfocus

A ação **setfocus** é utilizada para posicionar o cursor em um determinado campo, disparando um evento *xforms-focus* no elemento alvo. Um exemplo pode ser visto na

linha de comando abaixo, onde um evento de inicialização de formulário dispara a ação de posicionar o foco no controle de endereço.

```
<xforms:setfocus control="endereco" ev:event="xforms-initialize-done"/>
```

Ação send

Em algumas situações pode ser necessário submeter programaticamente o formulário, o que pode ser realizado com esta ação. É necessário identificar o elemento de submissão que será enviado. Um exemplo pode ser visto na linha de comando abaixo:

```
<xforms:send submission="submitForm1" ev:event="DOMActivate"/>
```

Ação reset

A ação **reset** é uma das mais comumente encontradas em formulários na Internet. No XForms, deve-se especificar a instância de modelo que será resetada. Na linha de comando abaixo pode-se ver um exemplo de ação **reset**.

```
<xforms:reset model="dadosCliente"  ev:event="DOMActivate"/>
```

Ação load

A ação **load** é utilizada para carregar algum outro recurso, substituindo o formulário atualmente exibido no navegador, ou abrindo em uma nova janela. Na linha de comando abaixo podemos ver um exemplo de ação **load**, disparada como resposta a um evento, que substituirá o formulário atualmente sendo visualizado no navegador.

```
<xforms:load resource="http://www.exemplo.com" ev:event="DOMActivate" show="replace"/>
```

Ação toggle

Ao utilizar elementos *switch* em XForms, deve-se especificar alguma forma de alternar os diferentes casos que podem ser visualizados. A alternância dos casos é realizada através da ação **toggle**, como visto no exemplo da linha de comando abaixo:

```
<xforms:toggle case="principal" ev:event="DOMActivate"/>
```

Ação insert

A ação **insert** trabalha em conjunto com um conjunto de nodos, construindo novos nodos a partir de um modelo. Deve-se especificar a localização dentro do conjunto aonde será inserido o novo elemento. Um exemplo pode ser visto na linha de comando abaixo:

```
<xforms:insert nodeset="enderecos/endereco" at="cursor('end')" position="after"/>
```

Ação delete

Ao contrário da ação **insert**, esta ação remove um elemento de um conjunto de nodos, indicado pelo atributo *at*. Um exemplo da ação **delete** pode ser visto na linha de comando abaixo:

```
<xforms:delete nodeset="enderecos/endereco" at="cursor('end')"/>
```

Ação setindex

Também usada em conjunto de nodos, a ação **setindex** mantém um apontador para o item corrente, da mesma forma que um índice. Deve-se especificar o identificador do elemento repetitivo, e o índice corrente desejado. Um exemplo pode ser visualizado na linha de comando abaixo:

```
<xforms:setindex repeat="enderecos" index="1" ev:event="DOMActivate"/>
```

Ação dispatch

Com a ação **dispatch** é possível programaticamente disparar um evento para qualquer elemento XForms. Deve-se especificar o nome do evento, o elemento alvo, se o evento passará pela fase de *bubbling* e se pode ou não ser cancelado (os dois últimos são atributos opcionais). Um exemplo pode ser visto na linha de comando abaixo:

```
<xforms:dispatch name="xforms-refresh" target="dadosCliente" ev:event="DOMActivate"/>
```

4.5.4 Tipos de Dados no padrão XForms

Os tipos de dados no padrão XForms estão baseados nos tipos de dados definidos para o XML Schema (W3C, 2003). De fato, apenas o tipo de dado *xsd:duration* não está disponível para o XForms.

O XML Schema define alguns tipos de dados primitivos, que podem ser estendidos para a criação de novos tipos de dados. Através do uso de facetas, pode-se especificar restrições aos tipos de dados primitivos, criando-se desta forma um subconjunto de valores válidos para um novo tipo de dados. As facetas disponíveis podem ser visualizados na Tabela 4.4.

Tabela 4.4: Facetas no XML Schema

Faceta	Descrição
enumeration	Especifica uma lista de valores válidos
fractionDigits	Especifica o número de casas após a vírgula
length	Especifica o tamanho exato em caracteres, ou bytes para tipos binários
maxExclusive	Especifica o valor máximo que não pode ser alcançado
maxInclusive	Especifica o valor máximo que pode ser alcançado
maxLength	Especifica o número máximo de caracteres, ou bytes para tipos binários
minExclusive	Especifica o valor mínimo que não pode ser alcançado
minInclusive	Especifica o valor mínimo que pode ser alcançado
minLength	Especifica o número mínimo de caracteres, ou bytes para tipos binários
pattern	Especifica uma expressão regular. É a faceta mais poderosas e usadas em tipos de dados no XML Schema
totalDigits	Especifica o número total de dígitos significantes
whiteSpace	Especifica como devem ser manipulados os espaços em branco

Um exemplo de uso de faceta para a definição de um novo tipo de dados baseado em strings pode ser visto na Figura 4.16

```
<xs:simpleType name="tipoNome">
    <xs:restriction base="xsd:string">
        <xs:maxLength value="80"/>
    </xs:restriction>
</xs:simpleType>
```

Figura 4.16: Exemplo de Uso de Faceta

Vários tipos de dados pré-definidos estão disponíveis, tanto na definição do XML Schema quanto no padrão XForms. Para uma lista detalhada dos tipos, sugere-se a leitura da definição do XML Schema (W3C, 2003) e da obra de Dubinko (2003).

4.5.5 Submissão de Dados no padrão XForms

Uma vez que os campos do formulário forem preenchidos, tipicamente o usuário submeterá o formulário para ser processado em algum dispositivo. Em termos do padrão XForms, um evento **xforms-submit** será disparado para o elemento **submission**, quando o usuário clicar no botão para submeter, ou até mesmo programaticamente, pelo desenvolvedor, em resposta a algum outro evento disparado no modelo.

Algumas atividades são disparadas, de forma a garantir que a submissão do formulário tenha sucesso. Diferentemente aos formulários HTML, a submissão em XForms garante a integridade do formulário.

O padrão XForms garante que não ocorram submissões duplicadas do mesmo formulário. Em formulários HTML, tipicamente isto ocorre quando o usuário clica duas vezes no botão para submeter, ou clica no botão de atualizar a tela. O desenvolvedor HTML precisa tratar essas ocorrências. No padrão XForms, isto é automaticamente gerenciado.

Os dados do formulário precisam ser validados. Uma das atividades disparada pelo processo de submissão é a validação de todos os campos, interrompendo o processo se qualquer um dos campos não estiver corretamente preenchido.

Uma outra atividade é garantir que os dados tenham sido enviados ao servidor com sucesso, evitando-se a perda de dados em situações como o servidor estar fora do ar. O padrão XForms garante a integridade nestes casos, já que o processo de submissão não terá sido completado.

O padrão XForms permite ainda especificar o que submeter. É possível especificar que partes do modelo serão enviadas para o servidor, evitando-se o envio de dados não essenciais ao processamento do formulário. Isto é bastante comum, por exemplo, para dados temporários, usados para cálculos intermediários no formulário. Não há necessidade de submetê-los para o servidor, uma vez que são úteis apenas para a apresentação ao usuário

A submissão tipicamente será realizada para uma localização, determinada no padrão XForms por uma expressão URI. A primeira parte da URI informa o protocolo, ou esquema, sendo tipicamente utilizado os protocolos “http” ou “https”. A segunda parte da URI especifica o endereço do servidor ou local onde o processamento será realizado.

Uma vez que o formulário tenha sido submetido com sucesso, o conteúdo da página pode ser inteiramente substituído por algum outro conteúdo (por exemplo, um agradecimento), ou apenas a instância pode ser substituída, ou mesmo a resposta inteira ser descartada. Futuramente o padrão XForms pode apresentar outras opções para o pós-submissão.

4.6 Implementações do padrão XForms 1.0

Nesta seção, vamos apresentar algumas das implementações já existentes para o padrão XForms 1.0. A maioria dos processadores existentes implementam parcialmente a especificação. Para cada implementação, serão feitas uma apresentação do histórico da ferramenta e uma descrição de seu funcionamento.

4.6.1 A Implementação X-Smiles

O X-Smiles (2003) é um navegador XML feito em Java, especificado para funcionar em computadores pessoais e demais dispositivos ligados em rede, com o objetivo principal de suportar serviços multimídia.

Iniciado como um projeto de estudantes da universidade de tecnologia de Helsinki, no laboratório de softwares de telecomunicação e multimídia, tornou-se uma iniciativa de código aberto no início do ano de 2001.

Os objetivos principais e secundários do X-Smiles são:

- **Desenvolver um navegador XML em Java**. O navegador deve ser escrito em Java puro, capaz de exibir documentos em vários dialetos XML, entre os quais: XSLT, XSL FO, SMIL e SVG.
- **Ser capaz de rodar em pequenos dispositivos**. Pequenos dispositivos que rodem a máquina virtual Java devem ser capazes de executar o X-Smiles.
- **Mesclar diferentes dialetos XML em um único documento**. De acordo com os espaços de nomes XML, o X-Smiles deve suportar diferentes dialetos no mesmo documento. Deve ser possível, por exemplo, misturar elementos do XML Schema com XForms.
- **Suportar conteúdo multimídia e *streaming***. O X-Smiles deve suportar conteúdo multimídia (áudio e vídeo), e suportar *streaming*.
- **Suportar serviços interativos através de formulários**. Neste tópico entra o XForms, como base para a implementação de formulários no X-Smiles
- **Suportar o EcmaScript**. O X-Smiles deve suportar a implementação de scripts em EcmaScript, para automação de páginas.

O X-Smiles, na sua versão mais atual (0.8), implementa parcialmente o padrão XForms 1.0. A descrição das funcionalidades XForms suportadas pelo X-Smiles pode ser consultada no site do X-Smiles (2003).

O X-Smiles foi completamente escrito em Java 2, permitindo desta forma a independência de plataformas. Devido a esta característica, o X-Smiles já foi testado em computadores de mão (*hand-helds*), televisão digital e computadores pessoais, embora ainda não utilizado em tais dispositivos em conjunto com o padrão XForms.

Pelos testes realizados, o X-Smiles mostrou-se uma implementação bastante estável. Possivelmente, isto se deve ao fato de ter sido a primeira implementação do padrão XForms na forma de navegador. O laboratório de softwares de comunicação e multimídia da Universidade de Helsinki é co-autor do padrão XForms 1.0.

4.6.2 A Implementação IBM XML Forms Package

O IBM XML Forms Package (JORDAN, 2002) é um conjunto de ferramentas e componentes com o objetivo de apresentar as possibilidades do padrão XForms. O conjunto é composto de dois módulos, o componente de modelo de dados, e o componente cliente.

O componente de modelo de dados provê um conjunto de interfaces Java para a criação, acesso e modificação de modelos de dados XForms. Disponibiliza também uma

tag library para JSP para comunicação com o modelo de dados, de forma a facilitar o trabalho do desenvolvedor.

O componente cliente inclui duas tecnologias: um controle ActiveX processador XForms e um compilador XForms em Java.

O controle ActiveX funciona embutido no navegador Internet Explorer, possibilitando aos desenvolvedores distribuírem documentos XHTML que usem o espaço de nomes do padrão XForms. O controle ActiveX reconhece os elementos do padrão XForms e gera uma página HTML que implementa o formulário, através de scripts. Esta implementação mostrou-se bastante estável e funcional durante os testes.

O compilador do padrão XForms em Java provê uma interface para compilar XHTML com elementos XForms, gerando HTML com scripts, da mesma forma que o controle ActiveX. Diferentemente do controle ActiveX, esta solução pode ser usada em navegadores diferentes do Internet Explorer.

Da mesma forma que o X-Smiles, o IBM XML Forms Package não implementa o padrão XForms em sua totalidade. Entretanto, mostrou-se uma implementação bastante estável. Este pacote é considerado pela IBM como uma avaliação de uma tecnologia emergente.

4.6.3 A Implementação OXF

O OXF (Open XML Framework) é um ambiente de processamento de documentos XML, baseado em J2EE, com o objetivo de disponibilizar aos desenvolvedores uma plataforma de desenvolvimento sólida e consistente. O OXF foi inicialmente desenhado para suportar o desenvolvimento de aplicações na Internet, mas pode ser utilizado em aplicações de processamento XML puro.

A plataforma OXF disponibiliza diversas tecnologias, entre elas o padrão XForms (rascunho de agosto de 2002), para proporcionar ao desenvolvedor a agilidade que necessita. O OXF é uma solução comercial da Orbeon (2003).

Da mesma forma que no IBM XML Forms Package, o OXF traduz o documento no padrão XForms para formulários HTML. A tradução é realizada no servidor OXF, durante a requisição da página pelo navegador. Nos testes realizados, o OXF pareceu ser uma implementação bastante estável, embora exista uma restrição de tempo para avaliação, devido à característica comercial do produto.

4.6.4 A Implementação Chiba

O Chiba (TURNER, 2001) é uma implementação em código aberto do padrão XForms. Implementa a maioria das seções da recomendação candidata, disponibilizando as funcionalidades do padrão XForms nos navegadores atualmente existentes.

A implementação das regras do padrão XForms é realizada inteiramente no lado servidor, o que significa que não é necessário que o navegador disponha de quaisquer recursos de linguagens de scripts no lado cliente.

O Chiba foi desenvolvido baseado em Java 1.2, buscando a sua distribuição em múltiplas plataformas. Utiliza XSLT para transformar documentos no padrão XForms na linguagem de marcação desejada (HTML, WML, VoiceML, etc.).

A distribuição atual do Chiba é baseada em Java Servlets, e pode rodar opcionalmente sobre a plataforma Apache Cocoon.

4.6.5 A Implementação FormsPlayer

O FormsPlayer (2003) é uma implementação do padrão XForms da X-Port, realizada durante a execução de um projeto para um cliente. A maioria das transações do projeto envolviam a exibição e manipulação de formulários, o que levou a implementação de um controle ActiveX para a manipulação de formulários no padrão XForms.

O FormsPlayer funciona apenas em navegadores Internet Explorer versão 6 ou superior, através do uso de espaços de nomes que serão manipulados pelo controle ActiveX disponibilizado. Com o uso do espaço de nomes, as definições do padrão XForms podem ser colocadas diretamente no documento XHTML. O código resultante é limpo, sem nenhum tipo de tradução para HTML com Scripts, já que a manipulação dos elementos do padrão XForms é realizada apenas pelo controle ActiveX.

O FormsPlayer é uma implementação parcial do padrão XForms, e atualmente está disponibilizado para download gratuito na sua versão beta.

4.6.6 A Implementação XFE

O XFE é uma implementação comercial da E-XML Media (2003), baseada no rascunho de janeiro de 2002 do padrão XForms. Consiste de um Applet Java, carregado no navegador Internet Explorer, que interpreta o documento no padrão XForms, interage com o usuário, e envia instâncias de documentos XML validados ao servidor.

Da mesma forma que nas demais implementações, o XFE implementa parcialmente as construções e eventos do padrão XForms.

4.6.7 A Implementação Xero

O Xero é uma implementação do padrão XForms da “Type.A” software, baseado na recomendação candidata do padrão XForms. Disponibiliza um módulo editor de formulários e a *engine* XForms.

O módulo editor de formulários é um ambiente de desenvolvimento visual para formulários, suportando a autoria de formulários baseada em esquemas. Possibilita diversas visualizações sobre o mesmo documento de trabalho. Durante os testes, mostrou-se um ambiente ainda instável, mas com muitas funcionalidades para o design de formulários no padrão XForms.

A *engine* do Xero é um processador do padrão XForms, baseada em um controle ActiveX. Executa apenas sobre navegadores Internet Explorer, e implementa a maioria das construções no padrão XForms. Funciona da mesma forma que o FormsPlayer, definindo através de um espaço de nomes o controle ActiveX que irá manipular os elementos do padrão XForms.

4.6.8 Escolha da Implementação para as Aplicações Piloto

Todas as implementações do padrão XForms atualmente existentes implementam parcialmente o modelo. Além disso, todas são ferramentas relativamente novas, muitas estando ainda em estágios iniciais de testes. Durante os testes realizados, diversos problemas de estabilidade foram detectados.

A implementação mais estável, e ainda com o diferencial de ser um projeto de código aberto, é o X-Smiles (2003). A escolha do X-Smiles deve-se ao fato de ser a ferramenta que possibilite a execução das aplicações pilotos projetadas para este trabalho.

5 DEFINIÇÃO DAS APLICAÇÕES PILOTO

A verificação do padrão XForms será realizada através de dois diferentes experimentos, com o objetivo de verificar os pontos fortes, pontos fracos, lições aprendidas e dificuldades durante a aplicação do padrão em uma situação real.

O foco da verificação será uma análise comparativa da utilização do padrão em relação à utilização de formulários HTML quanto aos aspectos listados nos objetivos deste trabalho.

Neste capítulo serão apresentadas as aplicações pilotos em relação aos requisitos e recursos implementados. Serão explanadas as formas de implementação escolhidas para cada um dos problemas envolvidos.

5.1 Aplicação Piloto 1

A primeira aplicação piloto será um cadastro de clientes, com alguns requisitos principais:

- **A aplicação deverá possibilitar a visualização das informações do cliente sem que seja necessário o uso de barras de rolagem**. A quantidade de informações a serem exibidas é relativamente grande, não cabendo em uma única tela. Será necessário utilizar algum recurso para melhorar a interface com o usuário
- **Um cliente pode possuir múltiplos endereços e múltiplas formas de contato**. Múltiplos endereços compreendem endereços comercial, residencial, casa na praia, entre outros. Múltiplas formas de contato significam que o cliente pode ser contatado de diversas formas: telefone comercial, telefone residencial, e-mail, pager, celular, etc.
- **Todos os campos do formulário devem apresentar dicas de preenchimento**. Este requisito procura facilitar o preenchimento do formulário por usuários inexperientes
- **Os tipos de endereços e tipos de formas de contato devem proporcionar fácil manutenção**. É comum adicionar-se uma nova forma de contato, ou um novo tipo de endereço. A aplicação deve proporcionar uma fácil manutenção no código para evitar um grande impacto quando isto for necessário.

Para atender estes requisitos, a aplicação piloto 1 fará a implementação de um formulário utilizando o padrão XForms. Com a aplicação piloto 1 está-se procurando

analisar a efetividade da solução que o padrão XForms proporciona para os aspectos descritos abaixo.

Múltiplas Etapas de Preenchimento

Para atender o requisito de evitar o uso de barras de rolagem, a aplicação piloto 1 fará uso do recurso de múltiplas páginas por formulário, proporcionado pelo padrão XForms, através dos elementos **switch** e **case**.

Formulários complexos possuem muitas vezes uma grande quantidade de campos a serem preenchidos. Para uma melhor experiência do usuário, normalmente quebra-se o preenchimento do formulário em etapas, sendo cada etapa uma página web distinta. O cadastro de cliente proposto implementa o preenchimento em múltiplas etapas.

A aplicação do cadastro de clientes conterá três etapas de preenchimento: Informações Básicas, Formas de Contato e Endereços.

Na etapa de informações básicas são preenchidos alguns dados essenciais do cadastro do cliente: Nome, CPF, RG, Data de Nascimento, Nome do Pai e Nome da Mãe.

Na etapa de Formas de Contato, serão informados os meios de contato com o cliente, com o respectivo identificador. Nesta etapa aparece pela primeira vez a característica dos elementos repetitivos, já que um cliente pode possuir diversos meios de contato. O usuário deverá informar o tipo do contato (telefone, celular, Pager, e-mail) e o número ou identificador.

Na etapa de Endereços, o usuário deverá informar os dados do endereço do cliente. Nota-se novamente a existência da característica da repetição, uma vez que o cliente pode possuir diversos endereços a ele relacionados. Os campos necessários para preenchimento nesta etapa são: CEP, Pais, Estado, Município, Bairro, Logradouro e Complemento.

Elementos Repetitivos

Alguns elementos de formulários podem possuir mais de uma instância. Um exemplo típico, no caso da aplicação piloto de cadastro de clientes, é a multiplicidade de elementos endereços para um mesmo cliente. Um cliente pode possuir múltiplos endereços comerciais e residenciais. Normalmente, este tipo de situação em formulários tradicionais é um problema bastante complexo de ser resolvido.

Para atender a necessidade de possibilitar o cadastramento de múltiplos endereços e múltiplas formas de contato, a aplicação piloto 1 fará uso do elemento de repetição disponível no padrão XForms, o elemento **repeat**.

Dicas de Preenchimento

O padrão XForms disponibiliza o elemento **hint** para a exibição de dicas rápidas de preenchimento de campos, sendo este elemento utilizado pela aplicação piloto 1.

Instâncias Auxiliares

Para facilitar a manutenção dos tipos de endereços e formas de contato, os mesmos serão disponibilizados em instâncias auxiliares dentro do mesmo modelo. O padrão

XForms permite a utilização de diversas instâncias buscando a simplificação do processo de manutenção do aplicativo.

Tipos de Dados no esquema XML

O padrão XForms utiliza os tipos de dados definidos no esquema XML para apresentar campos com controles adequados ao tipo de dado. A aplicação piloto 1 fará uso de campos de data, buscando verificar como o padrão XForms atende esta necessidade.

Cascatas de Estilo

Para prover uma interface com o usuário agradável, será utilizada na aplicação piloto 1 cascatas de estilo que visam modificar a aparência dos campos definidos no modelo. O padrão XForms prevê a integração com outras especificações.

Caixas de Seleção Unitárias

As caixas de seleção unitárias podem no padrão XForms serem vinculadas a instâncias auxiliares. Serão utilizadas na aplicação piloto 1 caixas de seleção unitária para o tipo de endereço e a forma de contato, vinculadas às instâncias auxiliares correspondentes.

Separação dos componentes de dados, visão e lógica

O padrão XForms procura se enquadrar dentro do padrão de projeto MVC. A aplicação piloto 1 será analisada quanto à efetividade da separação dos componentes.

Integração com outros conjuntos de elementos XML

A aplicação piloto 1 será implementada no padrão XForms, porém será embutida em uma página XHTML para visualização no navegador X-Smiles.

5.1.1 Descrição da Solução

A descrição da solução será feita por etapas. A primeira definição importante é o esquema XML utilizado. Em seguida, examinaremos o modelo e as instâncias XForms definidas para esta aplicação. O modelo de apresentação XForms, que define a apresentação do formulário propriamente dito, é explicitado, e finalmente o resultado será apresentado.

5.1.1.1 O Esquema XML

O esquema XML utilizado para a aplicação piloto 1 descreve a necessidade de negócio explicitada acima. O modelo gráfico do esquema pode ser visualizado na Figura 5.1.

A representação gráfica mostra os elementos utilizados na aplicação, e o relacionamento entre eles.

Um cadastro de cliente é composto por elementos simples de nome, CPF, RG e Data de Nascimento. Existe ainda um elemento composto por elementos para nome do pai e nome da mãe.

Pode-se verificar a existência de elementos repetitivos para as formas de contato e endereços do cliente, conforme especificado para a aplicação piloto 1. Cada forma de contato é composta por elementos para o tipo de contato e o identificador. Cada endereço é composto por elementos de tipo de endereço, CEP, País, Estado, Município, Bairro, Logradouro e Complemento.

O esquema XML, em forma textual, está representado na Figura 5.2.

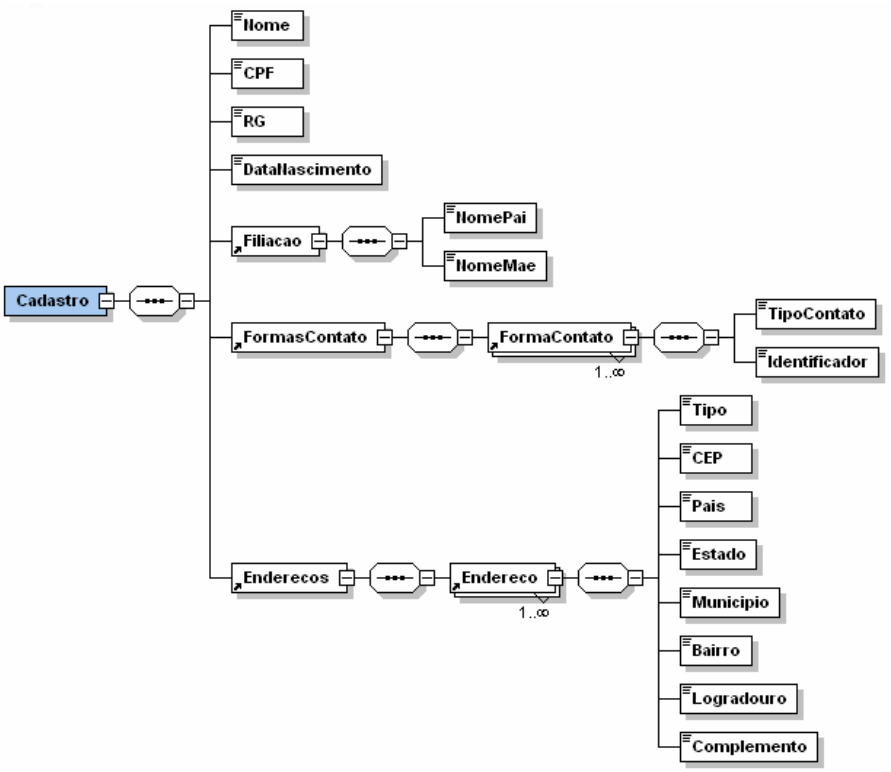


Figura 5.1: Modelo Gráfico do Esquema XML da Aplicação Piloto 1

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="Cadastro">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="Nome" type="xsd:string"/>
                <xsd:element name="CPF" type="xsd:string"/>
                <xsd:element name="RG" type="xsd:string"/>
                <xsd:element name="DataNascimento" type="xsd:date"/>
                <xsd:element ref="Filiacao"/>
                <xsd:element ref="FormasContato"/>
                <xsd:element ref="Enderecos"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="Filiacao">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="NomePai" type="xsd:string"/>
                <xsd:element name="NomeMae" type="xsd:string"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="FormasContato">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="FormaContato" maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="FormaContato">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="TipoContato" type="xsd:string"/>
                <xsd:element name="Identificador" type="xsd:string"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="Enderecos">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="Endereco" maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="Endereco">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="Tipo" type="xsd:string"/>
                <xsd:element name="CEP" type="xsd:string"/>
                <xsd:element name="Pais" type="xsd:string"/>
                <xsd:element name="Estado" type="xsd:string"/>
                <xsd:element name="Municipio" type="xsd:string"/>
                <xsd:element name="Bairro" type="xsd:string"/>
                <xsd:element name="Logradouro" type="xsd:string"/>
                <xsd:element name="Complemento" type="xsd:string"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

Figura 5.2: Esquema XML da Aplicação Piloto 1

5.1.1.2 O Modelo e as Instâncias XForms

A primeira definição importante em uma aplicação padrão XForms é o Modelo. O modelo pode especificar um esquema XML que será utilizado como padrão. Na aplicação piloto 1, o esquema *piloto1.xsd* é utilizado para a especificação da estrutura de dados e os tipos de dados de cada campo.

O elemento *submission* define para onde os dados serão enviados quando o formulário for submetido, e o protocolo que será utilizado para este fim. No exemplo, um arquivo XML será criado em um diretório com a instância gerada. A definição do modelo pode ser visualizada na Figura 5.3.

```
<xfm:model id="form1" schema="piloto1.xsd">
    <xfm:submission action="file://F:\teste.xml" id="submit1" method="put"/>
    <xfm:schema/>
```

Figura 5.3: Definição do Modelo da Aplicação Piloto 1

A primeira instância que discutiremos é a mais importante para a aplicação, já que será o repositório dos dados preenchidos pelo usuário. Esta instância deverá seguir a especificação do esquema proposto no modelo (no caso da aplicação piloto 1, *piloto1.xsd*).

A instância poderá ou não estar previamente preenchida com valores válidos. Isto permite a edição de um novo registro da aplicação, ou a alteração de um registro anteriormente existente. Pode-se ainda especificar um atributo *src* para o elemento *instance*, o que fará com que o processador XForms busque no endereço especificado a instância que deverá ser utilizada. Na Figura 5.4 pode-se visualizar a instância principal, que segue a estrutura imposta pelo esquema XML utilizado (*piloto1.xsd*).

```
<xfm:instance id="instance1" xmlns="">
    <Cadastro xmlns="" xmlns:ev="http://www.w3.org/2001/xml-events"
              xmlns:xfm="http://www.w3.org/2002/xforms/cr"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <Nome/>
        <CPF/>
        <RG/>
        <DataNascimento/>
        <Filiacao>
            <NomePai/>
            <NomeMae/>
        </Filiacao>
        <FormasContato>
            <FormaContato>
                <TipoContato/>
                <Identificador/>
            </FormaContato>
        </FormasContato>
        <Enderecos>
            <Endereco>
                <Tipo/>
                <CEP/>
                <Pais/>
                <Estado/>
                <Municipio/>
                <Bairro/>
                <Logradouro/>
                <Complemento/>
            </Endereco>
        </Enderecos>
    </Cadastro>
</xfm:instance>
```

Figura 5.4: Instância Principal da Aplicação Piloto 1

Instâncias auxiliares podem ser adicionadas ao modelo. Isto é extremamente útil para a especificação de informações auxiliares, como tabelas de códigos, de forma a facilitar a manutenção da aplicação. Na aplicação piloto 1 foram utilizadas duas instâncias auxiliares, uma para tipos de contatos e outra para tipos de endereços. Na Figura 5.5 pode-se visualizar as duas instâncias auxiliares utilizadas. Posteriormente, na especificação do modelo de apresentação do formulário, as informações das instâncias auxiliares serão utilizadas para preencher caixas de seleção.

```
<xfm:instance id="tiposContato">
    <temp xmlns="">
        <tiposContato>
            <option value="FONE">Telefone</option>
            <option value="CELULAR">Celular</option>
            <option value="EMAIL">E-Mail</option>
            <option value="PAGER">Pager</option>
        </tiposContato>
    </temp>
</xfm:instance>
<xfm:instance id="tiposEndereco">
    <temp xmlns="">
        <tiposEndereco>
            <option value="COM">Comercial</option>
            <option value="RES">Residencial</option>
            <option value="PRAIA">Casa da Praia</option>
            <option value="CAMPO">Casa no Campo</option>
        </tiposEndereco>
    </temp>
</xfm:instance>
```

Figura 5.5: Instâncias Auxiliares na Aplicação Piloto 1

5.1.1.3 O Modelo de Apresentação XForms

No modelo de apresentação XForms são definidos os elementos de interface com o usuário. A forma como o formulário será apresentado ao usuário é definida neste ponto.

Um dos requisitos para a aplicação piloto 1 é a quebra do preenchimento do formulário em etapas. No padrão XForms, isto é alcançado com a especificação de casos de seleção, em conjunto com gatilhos para a seleção do caso a ser exibido para o usuário.

Na Figura 5.6 pode-se ver a especificação dos gatilhos que disparam a exibição dos três diferentes aspectos da aplicação piloto 1: as Informações Básicas, as Formas de Contato e os Endereços. Os gatilhos são representados visualmente por botões que, ao serem clicados, mostram a etapa de preenchimento desejada. A especificação dos casos de visualização é feita através do elemento *switch*, que possuirá tantos elementos *case* quantas forem as etapas de visualização.

```
<xfm:trigger>
    <xfm:label>Informações Básicas</xfm:label>
    <xfm:toggle ev:event="xforms-activate" case="basico_show"/>
</xfm:trigger>
<xfm:trigger>
    <xfm:label>Formas de Contato</xfm:label>
    <xfm:toggle ev:event="xforms-activate" case="contato_show"/>
</xfm:trigger>
<xfm:trigger>
    <xfm:label>Endereços</xfm:label>
    <xfm:toggle ev:event="xforms-activate" case="endereco_show"/>
</xfm:trigger>
```

Figura 5.6: Gatilhos para Exibição dos Casos na Aplicação Piloto 1

Os casos de visualização são representados por elementos *case*, que explicitam um identificador único do caso. Todos os elementos filhos do caso serão exibidos ou não, conforme o gatilho acionado.

A definição da etapa de Informações Básicas da aplicação piloto 1 pode ser visualizada na Figura 5.7. Nesta etapa são especificados diversos campos de entrada (elementos *input*) para cada um dos campos que devem ser informados nesta etapa: Nome, CPF, RG, Data de Nascimento, Nome do Pai e Nome da Mãe.

Cada elemento *input* possui uma referência ao elemento na instância em que deve ser armazenado (atributo *ref*). Nota-se que a amarração à instância, que, por sua vez, está vinculada a um esquema XML que define tipos de dados para cada elemento, provoca a alteração na forma de visualização do elemento. Por exemplo, o elemento que representa a data de nascimento do cliente será visualizado na forma de um calendário.

```
<xfm:case id="basico_show">
    <xfm:group id="infoBasico">
        <p>
            <xfm:input ref="/Cadastro/Nome">
                <xfm:hint>Informe o seu nome completo neste campo</xfm:hint>
                <xfm:label>Nome:</xfm:label>
            </xfm:input>
        </p>
        <p>
            <xfm:input ref="/Cadastro/CPF">
                <xfm:hint>Informe o seu CPF neste campo</xfm:hint>
                <xfm:label>CPF:</xfm:label>
            </xfm:input>
        </p>
        <p>
            <xfm:input ref="/Cadastro/RG">
                <xfm:hint>Informe o seu RG neste campo</xfm:hint>
                <xfm:label>RG:</xfm:label>
            </xfm:input>
        </p>
        <p>
            <xfm:input ref="/Cadastro/DataNascimento">
                <xfm:hint>Informe a sua data de nascimento neste campo</xfm:hint>
                <xfm:label>Data de Nascimento:</xfm:label>
            </xfm:input>
        </p>
        <p>
            <xfm:input ref="/Cadastro/Filiacao/NomePai">
                <xfm:hint>Informe o nome do seu Pai neste campo</xfm:hint>
                <xfm:label>Nome do Pai::</xfm:label>
            </xfm:input>
        </p>
        <p>
            <xfm:input ref="/Cadastro/Filiacao/NomeMae">
                <xfm:hint>Informe o nome da sua Mãe neste campo</xfm:hint>
                <xfm:label>Nome da Mãe:</xfm:label>
            </xfm:input>
        </p>
    </xfm:group>
</xfm:case>
```

Figura 5.7: Etapa de Informações Básicas da Aplicação Piloto 1

A etapa de preenchimento de Formas de Contato do cliente é bem mais complexa, já que introduz o conceito de repetição de grupos de elementos, e a busca de dados em instâncias auxiliares para o preenchimento de caixas de seleção. A especificação XForms para esta etapa pode ser visualizada na Figura 5.8.

A característica de repetitividade é alcançada pelo elemento *repeat*, que trabalha sobre uma coleção homogênea de elementos endereçada pelo atributo *nodeset*. Através

de gatilhos que disparam operações de inserção (elemento *insert*) ou remoção (elemento *delete*), pode-se incluir ou remover elementos repetitivos na coleção.

Na inserção (elemento *insert*), o conjunto de elementos especificado na instância inicial, endereçado pelo atributo *nodeset* é utilizado para a inserção do novo elemento repetitivo homogêneo. O ponto de inserção é determinado pelos atributos *at* (que aponta algum elemento existente na coleção) e *position* (que determina se será inserido antes – *before* - ou depois – *after* - do elemento apontado).

Na remoção (elemento *delete*), o elemento a ser removido é localizado pelo atributo *nodeset*, que especifica qual a coleção de elementos alvo da operação, e pelo atributo *at*, que especifica a posição na coleção do elemento que deve ser removido.

Em ambos os casos, o posicionamento do apontador dentro da coleção é realizado pelo operador *index*, que especifica o elemento repetitivo corrente em uma coleção.

```
<xfm:case id="contato_show">
    <xfm:repeat id="listaContatos" nodeset="/Cadastro/FormasContato/FormaContato">
        <xfm:group id="infoContato">
            <p>
                <xfm:select1 ref="TipoContato">
                    <xfm:hint>Informe o tipo de contato neste campo</xfm:hint>
                    <xfm:label>Tipo de Contato:</xfm:label>
                    <xfm:itemset nodeset="instance('tiposContato')/tiposContato/option">
                        <xfm:label ref="."/>
                        <xfm:value ref="@value"/>
                    </xfm:itemset>
                </xfm:select1>
            </p>
            <p>
                <xfm:input ref="Identificador">
                    <xfm:hint>Informe o Identificador neste campo</xfm:hint>
                    <xfm:label>Identificador:</xfm:label>
                </xfm:input>
            </p>
        </xfm:group>
    </xfm:repeat>
    <xfm:trigger>
        <xfm:label>Nova Forma de Contato</xfm:label>
        <xfm:insert ev:event="xforms-activate"
                    position="after"
                    nodeset="/Cadastro/FormasContato/FormaContato"
                    at="xfm:index('listaContatos')"/>
    </xfm:trigger>
    <xfm:trigger>
        <xfm:label>Remover Forma de Contato</xfm:label>
        <xfm:delete ev:event="xforms-activate"
                     nodeset="/Cadastro/FormasContato/FormaContato"
                     at="xfm:index('listaContatos')"/>
    </xfm:trigger>
</xfm:case>
```

Figura 5.8: Etapa de Formas de Contato da Aplicação Piloto 1

O primeiro elemento dentro do grupo de repetição é uma caixa de seleção (elemento *select1*) para o tipo de contato. Para o tipo de contato, foi utilizada uma instância auxiliar para a especificação dos tipos de contatos existentes, conforme visto na Figura 5.5.

O preenchimento da caixa de seleção baseado na instância auxiliar é realizado através do elemento *itemset*, que especifica o conjunto de nodos (atributo *nodeset*) a ser utilizado para buscar os nodos da instância desejada. A partir desta instância, especifica-se os elementos *label* e *value* através do uso de expressões XPath.

A etapa de preenchimento de Endereços do cliente é tecnicamente similar à etapa de Formas de Contato. A característica da repetitibilidade é utilizada aqui novamente, assim como a instância auxiliar para determinar os tipos de endereços existentes. A especificação da etapa de preenchimento de endereços do cliente pode ser vista na Figura 5.9.

```
<xfm:case id="endereco_show">
    <xfm:repeat id="listaEnderecos" nodeset="/Cadastro/Enderecos/Endereco">
        <xfm:group id="infoEndereco">
            <p>
                <xfm:select1 ref="Tipo" appearance="minimal" selection="open">
                    <xfm:hint>Informe o tipo de endereço neste campo</xfm:hint>
                    <xfm:label>Tipo de Endereco:</xfm:label>
                    <xfm:itemset nodeset="instance('tiposEndereco')/tiposEndereco/option">
                        <xfm:label ref="."/>
                        <xfm:value ref="@value"/>
                    </xfm:itemset>
                </xfm:select1>
            </p><p>
                <xfm:input ref="CEP">
                    <xfm:hint>Informe o CEP neste campo</xfm:hint>
                    <xfm:label>CEP:</xfm:label>
                </xfm:input>
            </p><p>
                <xfm:input ref="Pais">
                    <xfm:hint>Informe o Pais neste campo</xfm:hint>
                    <xfm:label>Pais:</xfm:label>
                </xfm:input>
            </p><p>
                <xfm:input ref="Estado">
                    <xfm:hint>Informe o Estado neste campo</xfm:hint>
                    <xfm:label>Estado:</xfm:label>
                </xfm:input>
            </p><p>
                <xfm:input ref="Municipio">
                    <xfm:hint>Informe o Município neste campo</xfm:hint>
                    <xfm:label>Município:</xfm:label>
                </xfm:input>
            </p><p>
                <xfm:input ref="Bairro">
                    <xfm:hint>Informe o Bairro neste campo</xfm:hint>
                    <xfm:label>Bairro:</xfm:label>
                </xfm:input>
            </p><p>
                <xfm:input ref="Logradouro">
                    <xfm:hint>Informe o Logradouro neste campo</xfm:hint>
                    <xfm:label>Logradouro:</xfm:label>
                </xfm:input>
            </p><p>
                <xfm:input ref="Complemento">
                    <xfm:hint>Informe o Complemento neste campo</xfm:hint>
                    <xfm:label>Complemento:</xfm:label>
                </xfm:input>
            </p>
        </xfm:group>
    </xfm:repeat>
    <xfm:trigger>
        <xfm:label>Novo Endereço</xfm:label>
        <xfm:insert ev:event="xforms-activate"
                    position="after"
                    nodeset="/Cadastro/Enderecos/Endereco" at="xfm:index('listaEnderecos')"/>
    </xfm:trigger>
    <xfm:trigger>
        <xfm:label>Remover Endereco</xfm:label>
        <xfm:delete ev:event="xforms-activate"
                     nodeset="/Cadastro/Enderecos/Endereco" at="xfm:index('listaEnderecos')"/>
    </xfm:trigger>
</xfm:case>
```

Figura 5.9: Etapa de Endereços da Aplicação Piloto 1

O último elemento a ser explicado na aplicação piloto 1 é o elemento de submissão (*submit*), que apresenta-se como um botão na tela que, ao ser clicado, submete os dados preenchidos conforme o atributo de submissão (*submission*) especificado. O elemento de submissão pode ser visualizado na Figura 5.10.

```
<xfm:submit name="Submit" submission="submit1">
    <xfm:hint>Clique para Enviar</xfm:hint>
    <xfm:label>Enviar</xfm:label>
</xfm:submit>
```

Figura 5.10: O Elemento de Submissão na Aplicação Piloto 1

5.1.1.4 O Resultado Final da Aplicação Piloto 1

A apresentação final da aplicação piloto 1 pode ser vista na figura Figura 5.11

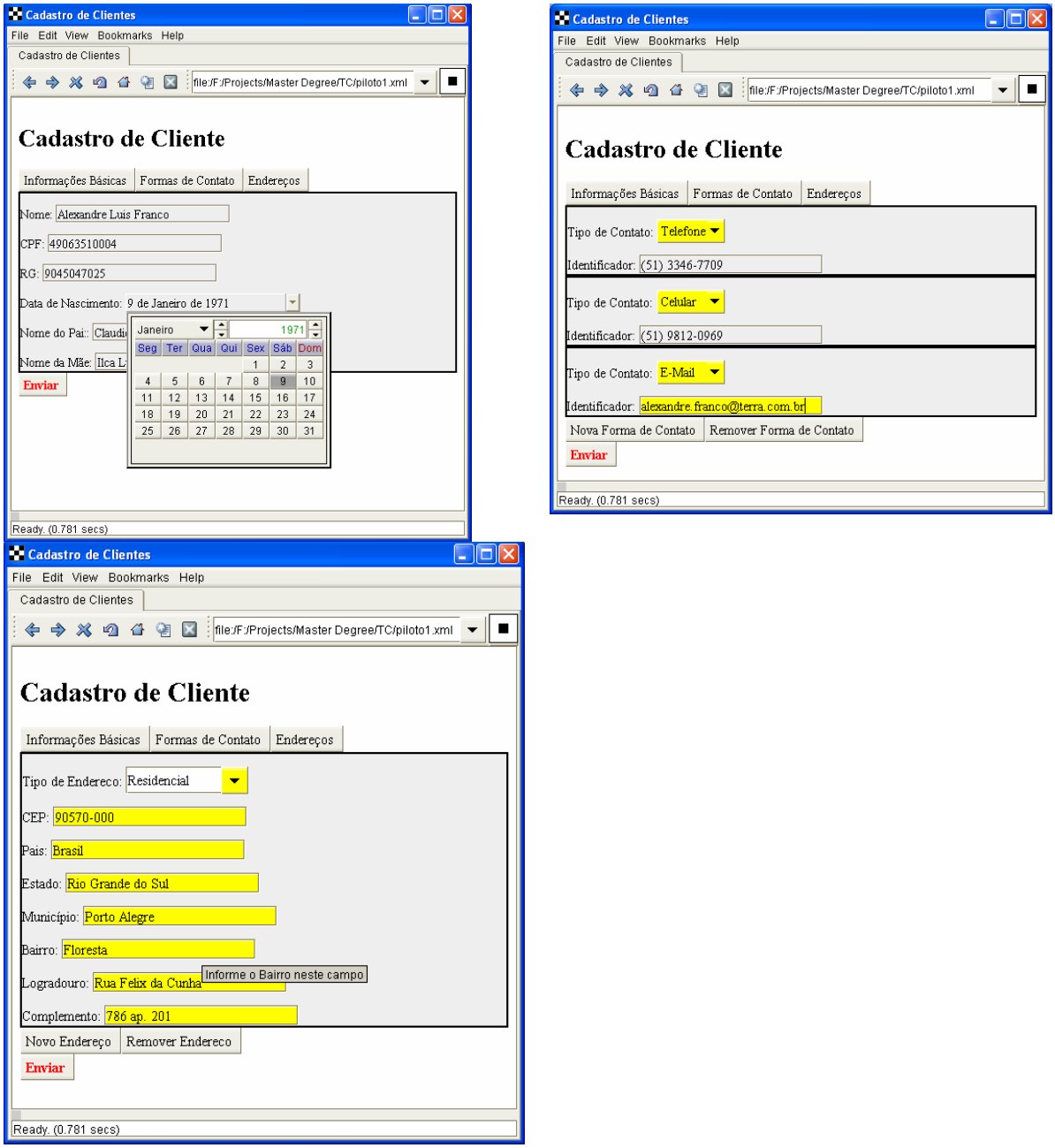


Figura 5.11: Apresentação Final da Aplicação Piloto 1

Percebe-se a forma como as múltiplas etapas foram endereçadas: cada etapa é acessada por botões no topo do formulário, conforme especificado no modelo XForms.

Nota-se ainda a apresentação do campo Data de Nascimento na forma de calendário. No modelo XForms (Figura 5.7), o campo Data de Nascimento está definido exatamente da mesma forma que os demais campos. Porém, o seu tipo de dado no esquema XML (*piloto1.xsd*) diz que o tipo de dados deste campo é uma data. A *engine* XForms utiliza esta definição para apresentar um controle adequado ao tipo de dados.

Um exemplo de instância, gerada pelo formulário especificado na aplicação piloto 1, pode ser visualizado na Figura 5.12

```
<?xml version="1.0" encoding="UTF-8"?>
<Cadastro xmlns="" xmlns:ev="http://www.w3.org/2001/xml-events"
xmlns:xfm="http://www.w3.org/2002/xforms/cr" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="F:\Projects\Master Degree\TC\piloto1.xsd">
    <Nome>Alexandre Luis Franco</Nome>
    <CPF>49063510004</CPF>
    <RG>9045047025</RG>
    <DataNascimento>1971-01-09</DataNascimento>
    <Filiacao>
        <NomePai>Claudio Gomes Franco</NomePai>
        <NomeMae>Ilca Lucia Franco</NomeMae>
    </Filiacao>
    <FormasContato>
        <FormaContato>
            <TipoContato>FONE</TipoContato>
            <Identificador>(51) 3346-7709</Identificador>
        </FormaContato>
        <FormaContato>
            <TipoContato>CELULAR</TipoContato>
            <Identificador>(51) 9812-0969</Identificador>
        </FormaContato>
        <FormaContato>
            <TipoContato>EMAIL</TipoContato>
            <Identificador>alexandre.franco@terra.com.br</Identificador>
        </FormaContato>
    </FormasContato>
    <Enderecos>
        <Endereco>
            <Tipo>RES</Tipo>
            <CEP>90570-000</CEP>
            <Pais>Brasil</Pais>
            <Estado>Rio Grande do Sul</Estado>
            <Municipio>Porto Alegre</Municipio>
            <Bairro>Floresta</Bairro>
            <Logradouro>Rua Felix da Cunha</Logradouro>
            <Complemento>786 ap. 201</Complemento>
        </Endereco>
        <Endereco>
            <Tipo>CAMPO</Tipo>
            <CEP>90000000</CEP>
            <Pais>Brasil</Pais>
            <Estado>Rio Grande do Sul</Estado>
            <Municipio>Glorinha</Municipio>
            <Bairro>Passo da Caveira</Bairro>
            <Logradouro>Estrada do Mato Grande</Logradouro>
            <Complemento>S/N</Complemento>
        </Endereco>
    </Enderecos>
</Cadastro>
```

Figura 5.12: Instância de Dados Exemplo Gerada pela Aplicação Piloto 1

5.2 Aplicação Piloto 2

A segunda aplicação piloto será uma extensão da aplicação piloto 1, adicionando três requisitos essenciais:

- **Campos Relacionados**. Na aplicação piloto 1 foram definidos campos para País, Estado e Município. Para a aplicação piloto 2, estes campos serão transformados para caixas de seleção unitária, sendo que existe uma relação bastante forte entre eles. Uma vez selecionado o País, a caixa de seleção de Estados será preenchida com os Estados existentes naquele País. Uma vez selecionado o Estado, a caixa de seleção de Municípios será preenchida com os Municípios existentes naquele País e Estado.
- **Internacionalização**. O formulário deverá possibilitar sua fácil tradução para diferentes idiomas.
- **Suporte a operações de Suspender / Resumir**. A aplicação piloto 2 deverá tentar implementar a operação de suspensão / resumo do preenchimento.

Para tentar atender estes requisitos, a aplicação piloto 2 fará uma extensão da aplicação piloto 1 utilizando o padrão XForms, buscando analisar a efetividade da solução que o padrão XForms proporciona aos seguintes aspectos.

Instâncias Auxiliares com Referências Externas

Referências externas serão utilizadas na aplicação piloto 2 visando atender ao requisito dos campos relacionados. A idéia consiste em carregar a instância auxiliar com as opções relativas ao campo relativo. O padrão XForms não disponibiliza uma técnica direta para atender esta necessidade, mas ela pode teoricamente ser suprida utilizando-se os recursos disponíveis no padrão.

Substituição de Conteúdo de Instâncias Auxiliares

Aliado ao recurso de instâncias auxiliares com referências externas, a substituição de conteúdo da instância permitiria a implementação dos campos relacionados e a integração com fontes de dados externas. A aplicação piloto 2 fará uso da substituição de conteúdo de instâncias em modelos separados do modelo principal, usando para isso métodos de submissão de formulários.

Elemento de Submissão

O elemento de submissão será utilizado na aplicação piloto 2 em conjunto com as instâncias auxiliares para a implementação do requisito de campos relacionados. A funcionalidade de substituição de conteúdo de instâncias será analisada durante a implementação da aplicação.

Instâncias Auxiliares no suporte à Internacionalização

O padrão XForms não disponibiliza nenhum recurso diretamente voltado para a internacionalização de formulários. Os recursos existentes podem ser usados para este

fim. Será analisada uma técnica para suportar múltiplos idiomas em uma aplicação utilizando-se instâncias auxiliares, e textos referenciando estas instâncias.

Suporte a Dados de Formulários Estruturados

A aplicação piloto 2 será integrada à fontes de dados externas, através do uso de referências externas, substituição de instâncias e elementos de submissão.

Suporte a Operações de Suspender / Continuar

Durante a implementação da aplicação piloto 2 será feita uma tentativa de implementação do recurso de suspensão / resumo de preenchimento de formulário, suportada pelo padrão XForms.

5.2.1 Descrição da Solução

Uma das características mais comuns em formulários são os campos relacionados. Campos relacionados (ou dependentes) são aqueles cujo conteúdo dependem do preenchimento prévio de algum outro campo no formulário. Um exemplo deste tipo de relacionamento é o campo de País, que, ao ser selecionado, define as opções do campo Estado. O campo Estado, por sua vez, ao ser selecionado, define as opções do campo Município.

Na Figura 5.13 pode-se visualizar o exemplo de campos relacionados para País, Estado e Município.

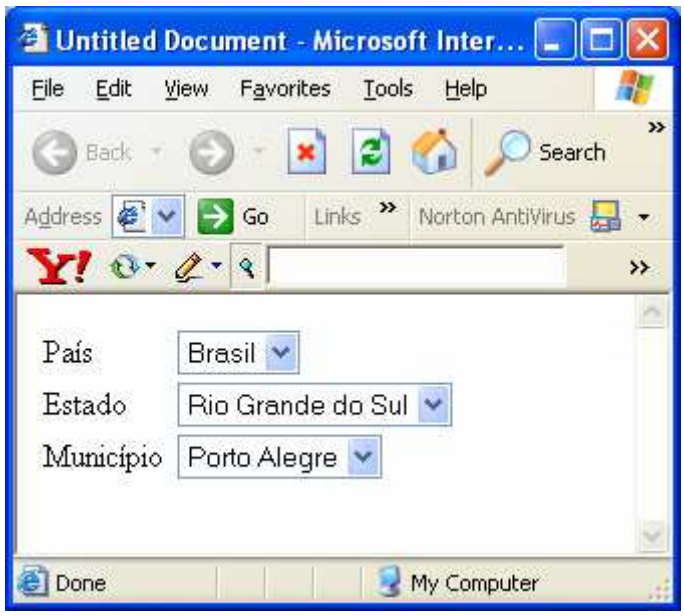


Figura 5.13: Exemplo de Campos Relacionados

A mudança proposta pela aplicação piloto 2 deverá ser realizada na etapa de preenchimento dos Endereços do cliente, substituindo-se os atuais campos de País, Estado e Município por caixas de seleção.

A segunda característica essencial que será adicionada refere-se à internacionalização do formulário, ou seja, a capacidade de manipular conjuntos de caracteres latinos e não-ocidentais. O formulário deve comportar-se independentemente do conjunto de caracteres utilizado.

Por último, o suporte a operações de suspensão / resumo de preenchimento será verificado. O formulário deve possibilitar a manutenção do estado de preenchimento do mesmo de alguma forma.

5.2.2 Campos Relacionados

O padrão XForms não implementa recursos com finalidade específica para manipulação de campos relacionados. Entretanto, uma forma para lidar com campos relacionados será apresentada nesta seção.

Campos relacionados são tipicamente implementados com caixas de seleção. Uma vez selecionada uma opção no primeiro campo, a segunda caixa de seleção é preenchida com as opções correspondentes à seleção feita pelo usuário. No padrão XForms, as caixas de seleção podem ser implementadas com elementos **select1**, como no exemplo da Figura 5.14.

```
<xfm:select1 ref="Pais">
      <xfm:hint>Informe o Pais neste campo</xfm:hint>
      <xfm:label>País:</xfm:label>
      <xfm:itemset nodeset="instance('Paises')/Paises/option">
            <xfm:label ref="."/>
            <xfm:value ref="@value"/>
      </xfm:itemset>
</xfm:select1>
```

Figura 5.14: Exemplo de Caixa de Seleção Unitária no XForms

A lista de opções que será apresentada ao usuário é definida em uma instância auxiliar, neste caso chamada de "*Paises"*. A definição da instância auxiliar pode ser vista na Figura 5.15.

```
<xfm:instance id="Paises">
        <temp xmlns="">
                <Paises>
                        <option value="BR">Brasil</option>
            <option value="EU">Estados Unidos</option>
        </Paises>
    </temp>
</xfm:instance>
```

Figura 5.15: Definição das instância auxiliar para a lista de Países

Da mesma forma que existe um elemento de seleção para País, e uma instância auxiliar para conter a lista de países, podemos criar elementos e instâncias para Estados e Municípios, que pode ser visto na Figura 5.16 .

```
<xfm:select1 ref="Estado">
    <xfm:hint>Informe o Estado neste campo</xfm:hint>
    <xfm:label>Estado:</xfm:label>
    <xfm:itemset model="Estados" nodeset="instance('Estados')/Estados/option">
    <xfm:label ref="."/>
    <xfm:value ref="@value"/>
    </xfm:itemset>
</xfm:select1>

<xfm:select1 ref="Municipio">
    <xfm:hint>Informe o Municipio neste campo</xfm:hint>
    <xfm:label>Município:</xfm:label>
    <xfm:itemset model="Municipios"
                nodeset="instance('Municipios')/Municipios/option">
    <xfm:label ref="."/>
    <xfm:value ref="@value"/>
    </xfm:itemset>
</xfm:select1>
```

Figura 5.16: Definição dos elementos select1 para campos Estado e Município

O padrão XForms mantém ativo o vínculo entre elementos. Quando a instância auxiliar é alterada em tempo de execução, a *engine* XForms automaticamente remonta a lista de opções que está vinculada a ela. Portanto, para remontar a lista de Estados pertencentes a um país, basta alterar a instância auxiliar que contêm as opções de Estados vinculado ao controle de seleção.

Instâncias no padrão XForms podem ser alteradas pela interação com o usuário. Neste caso, entretanto, não é uma forma adequada, já que a instância precisa ser redefinida completamente uma vez que o campo País tenha sido selecionado pelo usuário.

O padrão XForms possibilita que uma instância seja substituída pelo resultado de uma submissão realizada. Para isto, utiliza-se um elemento de submissão no modelo da instância auxiliar. Um exemplo desta abordagem pode ser visto na Figura 5.17.

```
<xfm:model id="Estados">
        <xfm:submission id="lookup_estados"
                        action="http://localhost:8080/estados.jsp"
                        method="get" replace="instance"/>
        <xfm:instance id="Estados">
                <selectedPais/>
                <Estados>
                        <option value=""></option>
                </Estados>
        </xfm:instance>
</xfm:model>
```

Figura 5.17: Modelo contendo instância auxiliar e elemento de submissão

No modelo auxiliar chamado “Estados”, além da definição da instância auxiliar temos agora um elemento de submissão chamado “lookup_estados”. Este elemento é responsável pela submissão do modelo para o endereço especificado no atributo *action*.

A chave para que a instância auxiliar seja substituída é o atributo replace=”instance” do elemento de submissão. Este atributo indica para o XForms que o resultado da submissão deve ser colocado como conteúdo da instância. Uma vez que a instância auxiliar tenha sido recarregada, a caixa de seleção de Estados é remontada.

No exemplo da Figura 5.17, a submissão será feita para a URI “http://localhost:8080/estados.jsp”. Esta URI é responsável por montar novas instâncias auxiliares para a caixa de seleção de estados.

Outro problema a ser resolvido é como informar à URI externa qual País deve ser utilizado de forma a criar as instâncias auxiliares de Estado corretamente. A solução adotada foi adicionar um elemento auxiliar na instância, chamado de **selectedPais**. Este elemento é submetido, juntamente com os demais elementos da instância, para a URI especificada. O conteúdo do elemento **selectedPais** contém o País atualmente selecionado na instância principal.

A definição da página “estados.jsp” pode ser vista na Figura 5.18. Trata-se de uma página Java Server Pages (JSP), cujo objetivo é montar instâncias XML de retorno de acordo com o País recebido como parâmetro. Caso o País seja o Brasil, a página retorna uma instância com os Estados do Rio Grande do Sul e Santa Catarina. Caso o País seja os Estados Unidos, retorna uma instância com os Estados do Arizona e Montana.

```
<%
String[] s = request.getParameterValues("selectedPais");

if (s[0].equals("BR")) {
%>
    <temp xmlns="">
        <Estados>
            <selectedPais>BR</selectedPais>
            <option value="RS">Rio Grande do Sul</option>
            <option value="SC">Santa Catarina</option>
        </Estados>
    </temp>
<%
} else if (s[0].equals("US")) {
%>
    <temp xmlns="">
        <Estados>
            <selectedPais>US</selectedPais>
            <option value="AR">Arizona</option>
            <option value="MO">Montana</option>
        </Estados>
    </temp>
<%
} else {
%>
    <temp xmlns="">
        <Estados>
            <selectedPais/>
            <option value=""></option>
        </Estados>
    </temp>
<% }; %>
```

Figura 5.18: Definição da Página “estados.jsp”

Estando o elemento de submissão definido, bem como o mecanismo de geração de instâncias (a página JSP), é necessário agora definir um mecanismo de atribuir o valor do país selecionado na instância principal com o elemento *selectedPais* na instância auxiliar. Isto é necessário toda a vez que o usuário selecionar um país na lista de opções.

O padrão XForms implementa um elemento chamado *bind*, que possibilita amarrar a nível de modelo dois campos de instâncias distintas. Um exemplo de *bind* pode ser visto na Figura 5.19. Este elemento vincula o País selecionado no modelo com o elemento *selectedPais* da instância auxiliar, possibilitando assim a sua submissão para a URI responsável por montar a lista de Estados do País selecionado.

```
<xfm:bind model=”Estados ref=”/selectedPais”
                         calculate=”/Cadastro/Enderecos/Endereço/Pais”/>
```

Figura 5.19: Exemplo de Bind para vincular dois campos no modelo

Uma outra forma de atribuir valor ao elemento *selectedPais* é através de uma ação na definição da interface com o usuário. Esta ação é definida pelo elemento *setvalue*, que programaticamente atribui um valor a um elemento de uma instância a partir de outro. Esta forma pode ser visualizada na figura Figura 5.20.

```
<xfm:trigger>
        <xfm:label>Lookup</xfm:label>
        <xfm:action ev:event="DOMActivate">
            <xfm:setvalue model="Estados" nodeset="/selectedPais" value="Pais"/>
             <xfm:send model="Estados" submission="lookup_estados"/>
        </xfm:action>
</xfm:trigger>
```

Figura 5.20: Exemplo de uso da ação setvalue

Para completar a implementação da solução para campos relacionados, é necessário definir uma forma para submissão da instância auxiliar para a URI responsável por montar a lista de Estados de um País. Utiliza-se uma ação *send*, que executa imediatamente a submissão definida pelo atributo *submission*. A ação *send* pode ser vista na Figura 5.21.

```
<xfm:trigger>
        <xfm:label>Lookup</xfm:label>
        <xfm:action ev:event="DOMActivate">
             <xfm:send model="Estados" submission="lookup_estados"/>
        </xfm:action>
</xfm:trigger>
```

Figura 5.21: Exemplo de uso da ação send

Cabe ressaltar que apesar de ambas formas de atribuir valores a elementos da instância serem suportadas pelo padrão XForms, a implementação do X-Smiles, em sua versão 0.81, utilizada no presente trabalho para validar os exemplos, não possibilitou a atribuição de valores por nenhum dos dois mecanismos. A validação da implementação foi realizada simulando o funcionamento da atribuição de valores.

5.2.3 Internacionalização do Formulário

Um dos objetivos-chave do padrão XForms é melhorar o suporte a aplicações que precisem rodar em diferentes conjuntos de caracteres. O padrão XForms entretanto, não dispõe de nenhum recurso específico para este fim. Dubinko (2003) propõe alguns padrões de formulários, entre eles, a localização XML, para atender esta necessidade.

Na aplicação piloto do cadastro de clientes, existe diversos textos que são passíveis de tradução em aplicativos multi - línguas. Tipicamente são identificados pelo elemento **label**, como pode ser visto na Figura 5.22.

```
<xfm:trigger>
        <xfm:label>Informações Básicas</xfm:label>
        <xfm:toggle ev:event="xforms-activate" case="basico_show"/>
</xfm:trigger>
```

Figura 5.22: Texto Passível de Tradução na Aplicação Piloto

O padrão Localização XML estabelece uma instância auxiliar para manter todos os textos utilizados pela aplicação. Na Figura 5.23 podemos ver a declaração de uma instância auxiliar apontando para textos na língua portuguesa.

```
<xfm:instance id=”strings” src=”http://localhost:8080/portugues.xml”/>
```

Figura 5.23: Instância auxiliar para textos na língua portuguesa

Tipicamente, esta instância auxiliar seria definida por uma URI externa, facilitando desta forma a manutenção dos textos bem como a exibição do formulário em uma outra língua: basta substituir o conteúdo do recurso apontado pelo atributo *src* da instância e o formulário será atualizado.

O conteúdo da instância define os termos que são utilizados na apresentação do formulário. Na Figura 5.24 podemos ver um exemplo de conteúdo para a instância, na língua portuguesa.

```
<strings>
        <botaoinfbasicas>Informações Básicas</botaoinfbasicas>
</strings>
```

Figura 5.24: Instância auxiliar de termos na língua portuguesa

Para usar os termos definidos pela instância auxiliar, basta substituir os valores fixos nos elementos de apresentação por referências à instância auxiliar. Um exemplo deste comportamento pode ser visto na Figura 5.25

```
<xfm:trigger>
        <xfm:label ref="instance('strings')/botaoinfbasicas"/>
        <xfm:toggle ev:event="xforms-activate" case="basico_show"/>
</xfm:trigger>
```

Figura 5.25: Utilização da Instância Auxiliar na camada de Apresentação

Desta forma, a substituição da instância auxiliar por uma que contenha os termos traduzidos para um outro idioma fará com que o formulário seja apresentado no idioma especificado. O mecanismo de localização fica restrito a uma instância auxiliar, não afetando as demais partes do formulário.

Apesar de previsto pelo padrão XForms, a implementação X-Smiles na sua versão 0.81 não possibilita a utilização de referências em elementos **label**, o que impossibilitou a verificação prática deste piloto.

5.2.4 Suporte a operações de Suspender / Resumir

O padrão XForms estabelece como um dos seus objetivos – chave o suporte a operações de suspender / resumir. A idéia é permitir ao usuário preencher o formulário aos poucos, sem perda dos dados já informados, e submeter o formulário completo quando desejar.

Este requisito está vinculado à implementação do padrão XForms, e não ao padrão XForms em si. Não existem construções ou elementos na especificação para suportar estas operações.

Alguns aspectos devem ser levados em conta quanto ao processo de suspender / resumir. Um formulário parcialmente preenchido precisa ser armazenado em alguma estrutura de dados no servidor ou na máquina cliente do usuário. Em ambos os casos, é necessário manter a identidade do usuário vinculado aos dados parciais do formulário. Além disso, não basta armazenar os dados parciais, mas também a referência para o tipo de formulário que está sendo utilizado, permitindo desta forma a continuação do seu preenchimento.

O X-Smiles na sua versão 0.81 não suporta preenchimento parcial do formulário, portanto não possibilita operações de suspender / resumir. A aplicação piloto portanto não pode verificar a utilização destas operações.

6 VERIFICAÇÃO DOS RESULTADOS DA APLICAÇÃO PILOTO

Neste capítulo serão levantados os resultados obtidos com a implementação da aplicação piloto. Os pontos fortes, pontos fracos, lições aprendidas e dificuldades serão explanados. Além disso, serão verificados os resultados quanto aos objetivos – chave do padrão XForms, e quanto aos problemas normalmente encontrados no desenvolvimento de aplicativos que utilizem formulários HTML.

6.1 Pontos Fortes

Nesta seção vamos analisar os pontos fortes encontrados na aplicação do padrão XForms para a solução do problema apontado na Aplicação Piloto.

6.1.1 Instâncias Auxiliares

A possibilidade de especificar instâncias auxiliares é um recurso extremamente poderoso para aplicações reais. Através deste recurso, pode-se especificar, em separado, todas as tabelas de dados auxiliares que são utilizadas, porém não são submetidas pela aplicação. Exemplos de uso para instâncias auxiliares são as tabelas de códigos, como as utilizadas na aplicação piloto 1 para tipos de formas de contato e tipos de endereço.

Combinada com a possibilidade de especificar-se fontes externas ao documento do formulário, através do atributo *src*, a aplicação XForms pode estabelecer interfaces com quaisquer outros aplicativos / bancos de dados legados existentes. Pode-se especificar como fonte externa, no atributo *src*, arquivos XML fisicamente localizados no servidor de aplicação. Pode-se ainda especificar URI's completas, desde que o resultado retornado seja um documento XML válido.

As instâncias auxiliares podem ser colocadas no mesmo modelo XForms que a instância principal, ou ser colocadas em modelos diferentes na mesma página. Podem ainda ter seu conteúdo inteiramente substituído durante o processamento de uma submissão de formulário – recurso este que foi utilizado para a implementação de campos relacionados na aplicação piloto.

Através das instâncias auxiliares implementou-se ainda a internacionalização do formulário. Todos os textos explicativos foram colocados em uma instância auxiliar, que pode ser carregada com diferentes traduções de acordo com a preferência do usuário.

6.1.2 Preenchimento em Etapas

A quebra do preenchimento de formulários em etapas, com o elemento *switch*, é bastante poderosa e fácil de ser implementada, como pode ser visto na aplicação piloto.

Aplicações que possuam uma grande quantidade de informações a serem preenchidas fatalmente precisam limitar o número de campos visíveis pelo usuário, para melhorar a usabilidade da aplicação. O padrão XForms possibilita esta quebra em etapas de uma forma bastante fácil e intuitiva.

6.1.3 Tipos de Dados

A definição dos tipos de dados, realizada no esquema XML relacionado ao modelo, pode mudar a forma de apresentação de um determinado campo na aplicação. Na aplicação piloto, isto pode ser verificado no campo Data de Nascimento, cujo tipo de dado no esquema XML foi definido como sendo uma data.

O resultado final é que o modelo XForms, como visto na Figura 5.7, torna-se bastante simplificado e limpo, não exigindo a mistura de informações semânticas ao modelo XForms.

6.1.4 Elementos Repetitivos

A Repetição de elementos é uma característica extremamente poderosa e simples de ser implementada no padrão XForms, como visto na aplicação piloto.

Tradicionalmente, formulários HTML apresentam uma dificuldade ímpar quando se tratam de elementos repetitivos. A inserção de mais um elemento repetitivo exige a reconstrução completa do formulário, o que normalmente se traduz em uma operação de postagem de dados ao servidor, para em seguida gerar um novo formulário com mais um elemento repetitivo a ser preenchido.

6.1.5 Elemento de Submissão

O elemento de submissão no padrão XForms dispõe de inúmeros recursos, alguns dos quais utilizados na aplicação piloto. Graças a ele, foi possível implementar os campos relacionados de País, Estado e Município, através da submissão de uma instância e sua posterior substituição pelo resultado do processamento.

6.2 Pontos Fracos

O principal ponto fraco detectado durante o desenvolvimento da aplicação piloto é quanto à estabilidade da implementação X-Smiles. Apesar de ainda ser uma das implementações mais estáveis, freqüentemente é necessário a reinicialização do

navegador. Além disso, as informações geradas para a depuração da aplicação são bastante limitadas, o que dificulta o trabalho do desenvolvedor. Alguns recursos do XForms estão parcialmente implementados, o que dificulta a verificação do padrão. A verificação da origem de um problema identificado pode ser confusa.

6.3 Lições Aprendidas

Nesta seção coletamos as lições aprendidas com o desenvolvimento da aplicação piloto no padrão XForms.

6.3.1 Definição do XML Schema

A definição do esquema XML a ser instanciado é a primeira etapa que deve ser realizada quando desenvolvendo uma aplicação que utilize o padrão XForms. Inicialmente, deu-se pouca ênfase na definição do esquema XML, o que trouxe diversos problemas posteriormente, durante a validação do modelo.

Um esquema bem definido contém, entre outras coisas, o tipo de dado de cada campo, o que é essencial para a validação do formulário. Define também a estrutura completa da instância que se deseja criar.

A implementação XForms é responsável por garantir a geração de instâncias que sigam fielmente o esquema XML definido. Entretanto, a utilização de um esquema XML introduz uma certa sobrecarga sobre o processamento da *engine* XForms, o que considera-se aceitável neste momento.

6.3.2 Elementos de Repetição

Os elementos de repetição são bastante poderosos, apesar de que o nível de abstração que deve-se utilizar com eles ser bastante alto. Não são elementos triviais, como os demais elementos do padrão XForms. Levou-se um tempo para compreender-se a forma como os elementos de repetição funcionam, pois assumem um conjunto de conceitos a respeito do modelo como sendo verdadeiros.

É necessário o perfeito entendimento dos conjuntos de nodos (*nodesets*) dentro dos elementos de repetição (*repeat*) e dento das operações de inserção (*insert*) e remoção (*delete*). A referência dos elementos de entrada (*inputs*, *select1*, entre outros) é sempre relativa ao *nodeset* do elemento *repeat*.

Nas inserções, o *nodeset* referencia o protótipo que será utilizado para inserção do elemento repetitivo, *conforme a instância do modelo inicialmente definida*.

Nas remoções, o *nodeset* determina a coleção homogênea que será alvo da operação de remoção, sendo o elemento desta coleção especificado pelo atributo *at* o elemento removido.

6.4 Dificuldades

A primeira dificuldade encontrada na aplicação piloto é quanto aos elementos de repetição, que possuem um entendimento não trivial em relação ao comportamento dos atributos *nodeset*, conforme visto na seção anterior.

A implementação de campos relacionados levou um certo tempo, visto que a implementação XForms escolhida para o desenvolvimento da aplicação piloto não contempla a atribuição de valores entre modelos distintos.

Uma outra dificuldade encontrada durante a implementação da aplicação piloto foi a identificação da origem de problemas que surgiram. Cada problema poderia ter como origem o padrão XForms em si ou a implementação do padrão escolhida para a aplicação piloto (no caso, o X-Smiles). A identificação da origem de cada problema exigiu pesquisas exaustivas nas referências bibliográficas deste trabalho.

O padrão XForms mostrou-se em geral bastante flexível, proporcionando diversas alternativas de implementação para determinados problemas. Essa flexibilidade, por outro lado, dificulta o trabalho do desenvolvedor, que precisa tomar a decisão de qual caminho seguir. O exemplo dos campos relacionados mostra diferentes abordagens que podem ser tomadas para obter-se o mesmo resultado.

6.5 Verificação Quanto aos Objetivos Chave do Padrão XForms

Nesta seção vamos repassar os objetivos chave do XForms, em relação ao escopo proposto para a aplicação piloto 1. A Tabela 6.1 apresenta os objetivos - chave XForms, e o status em relação à aplicação piloto 1.

Nota-se que dos nove objetivos – chave do XForms, a aplicação piloto 1 avaliou com sucesso seis, sendo que um não está no escopo deste trabalho.

6.5.1 Suporte a Handhelds, Televisão, e dektops, bem como impressoras e scanners

Como definido nos objetivos do presente trabalho, o suporte a dispositivos não desktop não foi verificado, já que não foram identificadas implementações do padrão XForms em plataformas distintas ao desktop.

6.5.2 Rica interface gráfica com o usuário, que atinja as necessidades de negócio, consumidores, e aplicações de controle de dispositivos

A aplicação piloto desenvolvida foi baseada na necessidade de cadastrar clientes de uma organização. Em relação à interface com o usuário, os seguintes aspectos foram verificados.

6.5.2.1 Múltiplas Etapas de Preenchimento – Switch e Case

Através dos elementos **switch** e **case**, pode-se facilmente adicionar etapas de preenchimento, como o realizado para a aplicação piloto. Na aplicação foram adicionadas três etapas, Informações Básicas, Formas de Contato e Endereços.

6.5.2.2 Elementos Repetitivos – Repeat

Através do elemento **repeat**, é possível implementar a entrada múltipla de tipos, como os realizados na aplicação piloto para as formas de contato e endereços. Cada cliente pode possuir inúmeras formas de contatos e endereços. A implementação de elementos repetitivos é trivial dentro da estrutura XForms

6.5.2.3 Folhas de Estilo em Cascata – CSS

O XForms possibilita a integração com cascatas de estilos – CSS, para definir uma interface com o usuário mais amigável. Este recurso foi amplamente utilizado na aplicação piloto 1 para definir estilos de fontes, cores, fundos e tamanhos de campos para os campos utilizados.

6.5.2.4 Controles Específicos para Tipos de Dados

O esquema XML utilizado para a aplicação piloto definiu tipos de dados para cada um dos seus campos. O resultado disso é que o XForms aproveitou-se desta informação e a utilizou para definir a melhor interface com o usuário possível para o tipo de dispositivo que se está utilizando, no caso, um desktop.

Por este motivo, o campo de data de nascimento é exibido em um formato de caixa de seleção calendário, que ao ser aberta, exibe um calendário para que o usuário selecione uma data.

6.5.2.5 Dicas de Preenchimnto – Hints

O XForms possibilita a utilização de dicas de preenchimento, que são exibidas quando o cursor do mouse paira sobre um determinado campo. Este recurso foi utilizado em todos os campos da aplicação piloto.

As dicas de preenchimento são recursos extremamente valiosos para a interface com o usuário, principalmente com usuários inexperientes.

6.5.2.6 Caixas de Seleção Unitárias – select1

A aplicação piloto utilizou amplamente o recurso de caixas de seleção unitárias, um recurso essencial para este tipo de aplicação. A integração com instâncias auxiliares

torna a caixa de seleção unitária um recurso extremamente poderoso no trato de campos relacionados em formulários.

6.5.3 Dados, Lógica e Apresentação Desacoplados

Na aplicação piloto, a definição das instâncias ficou completamente separada da definição da forma de apresentação do formulário. Esta abordagem facilitou a manutenção da aplicação, a inicialização de formulários e a manipulação de campos relacionados.

6.5.4 Internacionalização Melhorada

Através do padrão de formulários "Localização XML" proposto por Dubinko (2003), definiu-se uma instância auxiliar para manter todos os textos utilizados pela aplicação, ficando na definição da interface com o usuário apenas referências à instância de textos.

Como as instâncias podem ser carregadas de URI externas à aplicação, a substituição de um determinado idioma por outro é tão simples quanto substituir o conteúdo da instância que mantém os textos. A abordagem da substituição do conteúdo de instâncias é idêntica à utilizada para campos relacionados no formulário.

6.5.5 Suporte a Dados de Formulários Estruturados

A utilização de instâncias que apontem para origens de dados externas, através de URI's, possibilita, virtualmente, a interligação da aplicação escrita em XForms com qualquer sistema ou banco de dados legado que possa se comunicar em XML.

A submissão do formulário preenchido pode ser mapeada para métodos padrão de postagem de formulários http. Os dados podem posteriormente ser processados e armazenados em banco de dados estruturados.

6.5.6 Lógica de Formulários Avançada

A aplicação piloto estabeleceu o preenchimento do formulários em múltiplas etapas, uma para informações básicas, uma para formas de contato, e a última para endereços.

Além disso, recursos avançados de repetição de informações, como as definidas para as formas de contato e endereços, foram utilizadas.

Por último, campos relacionados, um problema tradicionalmente complexo em formulários HTML, foram implementados na aplicação piloto.

6.5.7 Múltiplos Formulários por Página, e Páginas por Formulários

A aplicação piloto implementou com sucesso o conceito de múltiplas páginas por formulário. Este recurso é extremamente importante para a definição de interfaces com o usuário amigáveis, principalmente para longos formulários.

6.5.8 Suporte a Operações de Suspender / Continuar

Como visto no capítulo anterior, o suporte a operações de suspender / continuar está relacionada à implementação XForms, e não ao padrão. O padrão não define elementos de construção para suportar essas operações, ficando a cargo do fabricante implementá-las na sua ferramenta.

O X-Smiles, na sua versão 0.81, utilizada para a implementação da aplicação piloto, não implementa o suporte às operações de suspender / continuar.

6.5.9 Fácil Integração com Outros Conjuntos de Elementos XML

A aplicação piloto integrou-se facilmente com outros conjuntos de elementos XML. Por se tratar de um dialeto XML, o XForms possui essa característica por natureza.

Em primeiro lugar foi utilizado um esquema XML específico para as informações que deveriam ser mantidas pela aplicação.

O espaço de nomes do XML Events foi utilizado por toda a aplicação, para vincular eventos a determinadas ações XForms.

Por último, a aplicação foi embutida em um documento XHTML, para poder ser apresentada no navegador X-Smiles.

6.5.10 Resumo dos Objetivos – Chave XForms Verificados na Aplicação Piloto

Na Tabela 6.1 pode-se visualizar uma tabela com o resumo dos objetivos – chave do XForms verificados na aplicação piloto.

Tabela 6.1: Resumo dos Objetivos – Chave XForms Verificados na Aplicação Piloto

	Objetivo Chave	Verificado	Comentários
N/D	Suporte a handhelds, televisão, e desktops, bem como impressoras e scanners	Não	Até o presente momento não foram identificadas implementações do padrão XForms em plataformas não PC, portanto o trabalho não irá avaliar este objetivo chave
OK!	Rica interface gráfica com o usuário, que atinja as necessidades de negócio, consumidores, e aplicações de controle de dispositivos	Sim	A interface gráfica da Aplicação Piloto foi implementada com sucesso, atingindo as necessidades de negócio especificadas na sua totalidade. Múltiplas etapas foram facilmente adicionadas ao formulário. A característica de repetitibilidade também foi adicionada com sucesso. Tipos de dados diferenciados, como datas, foram representados por controles adequados. CSS, dicas de preenchimento e caixas de seleção unitárias foram utilizadas para definir uma interface mais amigável.
OK!	Dados, lógica e apresentação desacoplados	Sim	Os dados na aplicação piloto ficaram totalmente isolados dentro do modelo XForms, uma seção separada da especificação da lógica e apresentação do formulário
	Internacionalização melhorada	Sim	Através de uma instância auxiliar, pode-se especificar todos os textos utilizados na aplicação. Essa instância pode ser então traduzida para qualquer idioma.
OK!	Suporte a dados de formulários estruturados	Sim	A possibilidade de especificar instâncias com fontes externas (atributos *src*) estabelece, virtualmente, a conectividade com qualquer aplicação ou banco de dados legado. O interfaceamento com fontes estruturadas ou não depende apenas da geração da informação no formato correto (XML), o que pode ser facilmente realizado.
OK!	Lógica de formulários avançada	Sim	A aplicação piloto estabeleceu o preenchimento do formulários em diferentes etapas. Algumas etapas definiram elementos repetitivos, que apresentam tradicionalmente dificuldades ímpares em aplicações tradicionais. Campos relacionados,

			que tradicionalmente apresentam uma alta complexidade, foram implementados.
OK!	Múltiplos formulários por página, e páginas por formulário	Sim	O preenchimento da Aplicação Piloto foi quebrado em múltiplas etapas, de forma bastante fácil e intuitiva, com o objetivo de melhorar a usabilidade da aplicação.
	Suporte a operações de suspender / continuar.	Não	Não Verificado na Aplicação Piloto.
OK!	Fácil integração com outros conjuntos de elementos XML	Sim	A Aplicação Piloto foi integrada com um esquema XML previamente definido para o cadastro de clientes. Além disso, utilizou-se o espaço de nomes do XML Events e XHTML.

6.6 Verificação dos resultados obtidos quanto aos problemas encontrados em formulários HTML

Nesta seção serão verificados os resultados obtidos com a implementação da aplicação piloto em relação aos problemas tradicionalmente encontrados em aplicações que utilizem formulários HTML tradicionais.

6.6.1 Mescla de Diferentes Componentes

Formulários HTML são resultantes de uma mistura de componentes modelo e visão. A proposta inicial do padrão XForms é separar logicamente estes dois conceitos.

O resultado obtido pela aplicação piloto mostra claramente a separação destes dois componentes. Por um lado, as instâncias, definidas dentro do modelo XForms. Por outro, a definição da apresentação com o usuário, referenciando elementos nas instâncias.

O XForms separa claramente o modelo da visão, resolvendo um dos problemas mais graves encontrados em formulários HTML.

6.6.2 Dependência de Dispositivos

Formulários HTML estão cada vez mais dependentes dos dispositivos na qual foram projetados para serem executados. A proposta do XForms era salvar a proposta da independência de dispositivos, sendo o formulário desenvolvido para qualquer plataforma.

A aplicação piloto não pode verificar esta proposta, uma vez que não foram identificadas até o momento implementações XForms em dispositivos fora a plataforma desktop.

6.6.3 Arquitetura de duas camadas

Aplicações que utilizem Formulários HTML possuem um processamento dito em duas camadas, ou seja, o formulário é preenchido no cliente, e submetido para o servidor onde é processado e formulário é destruído.

A aplicação piloto mostrou que o XForms possibilita a continuação da vida do formulário, mesmo após uma submissão. Este recurso foi utilizado para a implementação dos campos relacionados: uma instância de dados era submetida, sendo o resultado da submissão utilizado para substituir completamente o conteúdo da instância, podendo então o usuário continuar o preenchimento do formulário.

6.6.4 Dependência de Linguagens de Script

O desenvolvimento de aplicações baseadas em formulários HTML tradicionalmente é suportado por scripts na realização de cálculos e validações. O XForms propõe diversas formas para evitar o uso de scripts nas aplicações.

A aplicação piloto não necessitou de nenhum recurso de scripts para realizar a sua tarefa. Todas as funcionalidades foram implementadas por recursos embutidos no XForms, ou recursos disponibilizados por tecnologias relacionadas, como o XPath, o XHTML, etc.

6.6.5 Repetitividade e Retrabalho

Desenvolvedores de aplicações baseadas em formulários HTML estão acostumados a usar “bibliotecas” de código comuns, muitos deles são copiados, colados e adaptados para uma determinada situação.

Na aplicação piloto não foi possível verificar este tipo de necessidade, talvez pela imaturidade de uso na tecnologia.

6.6.6 Dificuldade de Inicialização de Formulários

A inicialização de formulários HTML tipicamente exige a construção de uma nova instância do formulário, ao contrário do padrão XForms onde basta se atribuir uma nova fonte de dados.

A inicialização de formulários pode ser facilmente visualizada na aplicação piloto, proporcionada pelas instância separadas do resto do formulário. O recurso de inicialização foi utilizado amplamente na implementação dos campos relacionados.

6.6.7 Formato de Codificação

Formulários HTML baseiam-se em capturar e enviar pares de nome / valor, enquanto o padrão XForms utiliza documentos XML para a estrutura de dados.

A aplicação piloto pode verificar a eficácia do formato de codificação, nos elementos repetitivos de formas de contato e endereços. Os elementos repetitivos são desafiadores para estruturas de pares nome / valor, mas para formatos de instâncias robustas como o XML, isto não causa maiores problemas.

7 CONSIDERAÇÕES FINAIS

O desenvolvimento de aplicações baseadas em Internet apresenta diversos desafios que precisam ser endereçados para a construção de softwares robustos. A evolução dos requisitos das aplicações que utilizam formulários HTML foi muito além do previsto, criando situações complexas que exigem recursos altamente capacitados. Vários situações complexas foram expostas neste trabalho.

Alguns trabalhos dispersos começaram a ser desenvolvidos de forma a minimizar os problemas encontrados nas aplicações Internet. Cada fabricante desenvolveu seu próprio modelo de funcionamento, criando pequenos nichos espalhados pelo mercado.

O W3C, conhecendo a necessidade crescente de um novo padrão para atender a demanda dos desenvolvedores, criou o padrão XForms, que já possui algumas implementações comerciais disponíveis. O padrão XForms é uma tecnologia emergente, que apresenta uma série de inovações para formulários na Internet.

Algumas das principais implementações do XForms foram apresentadas neste trabalho. A maioria está ainda nos estágios iniciais de distribuição, apresentando diversos erros e não implementando o padrão XForms em sua totalidade. Outras implementações possuem um histórico anterior ao padrão XForms, tendo sido adaptadas assim que o padrão XForms começou a ser discutido.

A imaturidade das implementações dificultou a execução da análise deste trabalho. Muitas das dificuldades encontradas foram de fato causadas pela implementação utilizada, e não pelo padrão XForms em si.

Através da implementação de duas aplicações piloto, pode-se observar o comportamento do padrão XForms frente a uma necessidade real. As aplicações piloto definiram requisitos os quais exigiram o uso de alguns recursos previstos no padrão. Outros requisitos não são contemplados diretamente pelo padrão XForms, exigindo a utilização de alguns artifícios mostrados no trabalho.

Entretanto, todos os requisitos das aplicações piloto foram contemplados, diretamente ou não, pelo uso do padrão XForms. Os pontos fortes, pontos fracos, dificuldades e lições aprendidas foram coletados durante a implementação das aplicações piloto, analisados e documentados neste trabalho.

Por fim, foi realizada uma análise dos resultados obtidos quanto aos objetivos – chave do padrão XForms, e quanto às dificultades e limitações encontradas no desenvolvimento de aplicações que utilizam formulários HTML.

O presente trabalho não analisou o padrão XForms em sua totalidade, mas sim, contextualizou o padrão em relação ao passado, presente e futuro das aplicações Internet, ao mesmo tempo coletando uma base de conhecimento para ser utilizada em futuras aplicações do padrão.

REFERÊNCIAS

ABITEBOUL, Serge; BUNEMAN, Peter; SUCIU, Dan. **Data on the Web: From Relations to Semistructured Data and XML**. San Francisco: Morgan Kaufmann, 2000.

ADOBE. **The Adobe Acrobat eForms Soultion**. 2001. Disponível em: <http://www.adobe.com/products/acrobat/pdfs/eforms_faq.pdf>. Acesso em: 11 maio 2003.

ADOBE. **Creating eForms Solutions with Adobe Acrobat 5.0**. 2001. Disponível em: <http://www.adobe.com/products/acrobat/pdfs/eforms_recipe.pdf>. Acesso em: 11 maio 2003.

ADOBE. **Adobe Acrobat eForms Solution**. 2001. Disponível em: <http://www.adobe.com/products/acrobat/pdfs/eforms_workgroup.pdf>. Acesso em: 11 maio 2003.

ADOBE. **Adobe Acrobat eForms**. 2003. Disponível em: <http://www.adobe.com/products/acrobat/eforms.html>. Acesso em: 11 maio 2003.

AMGRAF. **OneForm Designer Plus**. 2003. Disponível em: <http://www.amgraf.com/pages/oneformplus1.html>. Acesso em: 15 maio 2003.

BALSOY, O. **SchemaWizard and XML**. 2002. Disponível em: <http://ptlportal.ucs.indiana.edu/schemawizard/schemawizard-nov12.pdf>. Acesso em: 16 maio 2003.

BERNERS-LEE, T.; CONNOLLY, D. **Hypertext Markup Language 2.0**. 1995. Disponível em: <http://www.ietf.org/rfc/rfc1866.txt>. Acesso em: 5 maio 2003.

BRANNAN, Heidi-Marie. **XMLForm Wizard How-To**. 2002. Disponível em: <http://xml.apache.org/cocoon/howto/xmlform-wizard/howto-xmlform-wizard.html>. Acesso em: 28 abr. 2003.

BUSCHMANN, Frank et al. **Pattern-Oriented Software Architecture:** a System of Patterns. Chichester: John Wiley & Sons, 1996.

BUTLER, Mark et al. Device Independence and the Web. **IEEE Internet Computing**, Washington, v.6, n.5, p.81-86, Sept.-Oct. 2002.

CHAN, Daniel K. C. Form Management in the Vicomte Workflow System. In: ANNUAL HAWAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES, HICSS, 32., 1999. **Proceedings…** [S.l.]: IEEE, 1999. v.5, p.5057.

CONNOLLY, Daniel W.; BERNERS-LEE, Tim. **Hypertext Markup Language**. 1993. Disponível em: <http://www.w3c.org/MarkUp/1995-archive/html-spec.html>. Acesso em: 5 maio 2003.

CONNER, Craig. **Care and Feeding of the FDF Toolkit**. 2001. Disponível em: <http://www27.placeware.com/etc/pws/adobe//recordings-adobe/1uuookssp4vm0/20011018-4t9659/SlideLog.html#Anchor0>. Acesso em: 11 maio 2003.

CONNECTIS. **Implementing an e-Forms Warehouse**. 2003. Disponível em: <http://www.connectisgroup.com/cs_eforms.htm>. Acesso em: 15 maio 2003.

DTS. **E-Forms**. 2003. Disponível em: <http://www.dts-inc.com/services/eforms.html>. Acesso em: 15 maio 2003.

DUBINKO, Micah. **What Are XForms?** 2002. Disponível em: <http://www.xml.com/pub/a/2001/09/05/xforms.html>. Acesso em: 28 abr. 2003.

DUBINKO, Micah et al. **XForms 1.0 – W3C Candidate Recommendation**. 2002. Disponível em: <http://www.w3.org/TR/2002/CR-xforms-20021112/index-all.html>. Acesso em: 28 abr. 2003.

DUBINKO, Micah. **XForms Essentials**. Sebastopol: O'Reilly & Associates, 2003.

DUMAS, Marlon. Ontology Markup for Web Forms Generation. In: WWW WORKSHOP ON REAL-WORLD APPLICATIONS OF RDF AND THE SEMANTIC WEB, 2002, Honolulu, HI. **Proceedings…** New York: ACM Press, 2002.

E-XML MEDIA. **XFE Web Site**. Disponível em: <http://www.e-xmlmedia.com/prod/xfe.htm>. Acesso em: 22 Jun. 2003.

FAWCETTE. **XForms Benefit All Stakeholders**. 2003. Disponível em: <http://www.fawcette.com/xmlmag/2002_12/magazine/practice/atomasic/sidebar1.asp>. Acesso em: 28 abr. 2003.

FITCH, Kent. Schema Driven User Interface Generation. In: AUSTRALIAN WORLD WIDE WEB CONFERENCE, AUSWEB, 8., 2002. **Proceedings…** [S.l.:s.n.], 2002.

FORMATTA. **Formata EForms Overview**. 1997. Disponível em: <http://www.formatta.com/forms/eforms_overview.htm>. Acesso em: 11 maio 2003.

FORMSPLAYER. **FormsPlayer Web Site**. 2003. Disponível em: <http://www.formsplayer.com/>. Acesso em: 22 jun. 2003.

GAMMA, Erich et al. **Design Patterns**: Elements of Reusable Object-Oriented Software. Reading: Addison Wesley, 1994.

GOLDBERG, Adele. **Smalltalk-80**: The Interactive Programming Environment. Reading: Addison-Wesley, 1983.

HONKALA, Mikko. XForms in X-Smiles. In: INTERNATIONAL CONFERENCE ON WEB INFORMATION SYSTEMS ENGINEERING, WISE, 2., 2001. **Proceedings…** Kyoto, Japan: IEEE, 2001.

IBM. **IBM XML Forms Package**. 2003. Disponível em: <http://www.tawpi.org/today/IMG/oct_implementing_eforms.pdf>. Acesso em: 11 maio 2003.

INFOZONE. **Schemox**. 2003. Disponível em: <http://www.infozone-group.org/projects_main.html>. Acesso em: 15 maio 2003.

JORDAN, Eric. **Implementing e-forms in a paper world**. 2002. Disponível em: <http://www.alphaworks.ibm.com/tech/xmlforms>. Acesso em: 28 abr. 2003.

KOTEK, Brian. **MVC design pattern brings about better organization and code reuse**. 2002. Disponível em: <http://www.zdnet.com.au/builder/architect/ood/story/0,2000034910,20269556,00.htm>. Acesso em: 21 maio 2003.

LEUNG, Karl; CHUNG, Jojo. The Liaision Workflow Engine Architecture. In: ANNUAL HAWAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES, HICSS, 32., 1999. **Proceedings…** [S.l.]: IEEE, 1999. v.5, p. 5059.

LEUNG, Karl; HUI, Lucas. Multiple Signature Handling in Workflow Systems. In: ANNUAL HAWAII INTERNATIONAL CONFERENCE ON SYSTEMS SCIENCE, HICSS, 33., 2000. **Proceedings…** [S.l.]: IEEE, 2000. v.6, p. 6033.

LOGIFORMS. **Logiforms Web Site**. 2003. Disponível em: <http://www.logiforms.com>. Acesso em: 28 abr. 2003.

MAES, Stéphane H. A "Single Authoring" Programming Model: the Interaction Model. In: SYMPOSIUM ON APPLICATIONS AND THE INTERNET, SAINT, 2002. **Proceedings…** [S.l.]: IEEE, 2002. p. 12.

McCARRON, Shane et al. **XML Events**: an Events Syntax for XML. 2003. Disponível em: <http://www.w3.org/TR/xml-events/>. Acesso em: 19 jun. 2003.

MICHEL, Thierry. **XForms – the next generation of Web forms**. 2003. Disponível em: <http://www.w3.org/MarkUp/Forms/>. Acesso em: 28 abr. 2003.

NOVELL. **Novell XForms Technology Preview**. 2003. Disponível em: <http://www.novell.com/xforms>. Acesso em: 28 abr. 2003.

ORACLE. **FBS Design**. 2003. Disponível em: <http://otn.oracle.com/sample_code/tutorials/fbs/over/design.htm>. Acesso em: 21 maio 2003.

ORBEON. **XForms Processor**. 2003. Disponível em: <http://www.orbeon.com/oxf/doc/processors-xforms>. Acesso em: 28 abr. 2003.

Q&D. **WebForms Web Site**. 2001. Disponível em: <http://www.q-d.com/wf.htm>. Acesso em: 28 abr. 2003.

PEMBERTON, Steven et al. **XHTML 1.0 The Extensible Hypertext Markup Language. (2nd ed.)**. 2002. Disponível em: <http://www.w3.org/TR/xhtml1/>. Acesso em: 5 maio 2003.

PREE, Wolfgang. **Design Patterns for Object-Oriented Software Development**. Workingham: Addison-Wesley, 1995.

PUREEDGE. **PureEdge Designer**. 2003. Disponível em: <http://www.pureedge.com/products/products/designerfeatures.php>. Acesso em: 15 maio 2003.

RAGGET, Dave. **HTML+ (Hypertext Markup Language)**. 1993. Disponível em: <http://www.w3.org/MarkUp/HTMLPlus/htmlplus_1.html>. Acesso em: 5 maio 2003.

RAGGET, Dave. A Review of the HTML+ Document Format. In: INTERNATIONAL CONFERENCE ON THE WORLD-WIDE WEB, 1., 1994. **Proceedings…** [S.l.:s.n.], 1994. p. 5-14.

RAUSCH, Andreas. A Proposal for a Code Generator based on XML and Code Templates. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, Toronto, Canada. **Proceedings…** [S.l.:s.n.], 2001.

RIVERA, Joel; TAING, Len. **Get ready for XForms**. 2002. Disponível em: <http://www-106.ibm.com/developerworks/library/x-xforms/>. Acesso em: 28 abr. 2003.

SUN. **Model-View-Controller**. 2002. Disponível em: <http://java.sun.com/blueprints/patterns/MVC.html>. Acesso em: 21 maio 2003.

SUN. **Java BluePrints**: Model-View-Controller. 2002. Disponível em: <http://java.sun.com/blueprints/patterns/MVC-detailed.html>. Acesso em: 21 maio 2003.

TURNER, Joern. **Chiba Web Site**. 2001. Disponível em: <http://chiba.sourceforge.net/>. Acesso em: 28 abr. 2003.

TYPEA. **Xero Web Site**. 2003. Disponível em: <http://typeasoft.com/product/xero/ >. Acesso em: 28 abr. 2003.

VIERINEN, Juha; VUORIMAA, Petri. A Browser User Interface for Digital Television. In: INTERNATIONAL CONFERENCE IN CENTRAL EUROPE ON COMPUTER GRAPHICS, VISUALIZATION AND COMPUTER VISION, WSCG, 2001. **Proceedings…** [S.l.:s.n.], 2001. p.174-181.

VUORIMAA, Petri et al. A Java Based XML Browser for Consumer Devices. In: SYMPOSIUM ON APPLIED COMPUTING, 2002. **Proceedings...** [S.l.:s.n.], 2002.

W3C. **XForms 1.0**: Data Model. 2000. Disponível em: <http://www.w3.org/TR/2000/WD-xforms-datamodel-20000406/>. Acesso em: 25 maio 2003.

W3C. **XForms 1.0**: Data Model. 2000. Disponível em: <http://www.w3.org/TR/2000/WD-xforms-datamodel-20000815/>. Acesso em: 25 maio 2003.

W3C. **W3C Web Site**. 2003. Disponível em: <http://www.w3c.org>. Acesso em: 5 maio 2003.

W3C. **XHTML 2.0**. 2003. Disponível em: <http://www.w3c.org/TR/2003/WD-xhtml2-20030131/ >. Acesso em: 6 maio 2003.

W3C. **Extensible Markup Language (XML)**. 2003. Disponível em: <http://www.w3c.org/XML/>. Acesso em: 13 maio 2003.

W3C. **The Extensible Stylesheet Language (XSL)**. 2003. Disponível em: <http://www.w3c.org/Style/XSL/>. Acesso em: 13 maio 2003.

W3C. **Document Object Model**. 2003. Disponível em: <http://www.w3c.org/DOM/>. Acesso em: 13 maio 2003.

W3C. **XML Schema**. 2003. Disponível em: <http://www.w3.org/XML/Schema>. Acesso em: 15 maio 2003.

W3C. **XForms Working Group Charter**. 2003. Disponível em: <http://www.w3.org/MarkUp/Forms/2003/xforms-wg-charter.html>. Acesso em: 25 maio 2003.

X-SMILES. **X-Smiles Web Site**. 2003. Disponível em: <http://www.xsmiles.org/>. Acesso em: 22 jun. 2003.