BC Billing UI

for

Open Source Clinical Application Resource

Version 0.4

Atefeh Mirsafian
Mayssam Mohammadi Nevisi

March 5, 2011

Contents

Revision History

Version	Date	Comments
0.1	Feb 27, 2011	Document outline
0.2	Feb 28, 2011	chapter 1: project definition
0.3	March 1, 2011	chapter 2: automatic form generation and XForms
0.4	March 5, 2011	chapter 3: bc billing architecture

Chapter 1

Project Definition

1.1 Introduction

Our goal is to design a web-based user interface to allow the users of OSCAR Canada [1] write billing rules and advisories.

This document is organized as follows:

- Chapter 1 introduces the project and its definition.
- Chapter 2 discusses an approach to automatic web application development.
- Chapter 3 describes general architecture of the developed solution.

1.2 Project Definition

In this section, we briefly introduce the project.

1.2.1 About OSCAR

OSCAR stands for Open Source Clinical Application Resource. It is a suite of Web-based Applications which serve to enhance the health and social service community's ability to provide the highest level of care [1].

All OSCAR applications are built in free Open Source software, which means you are free to download the program, use the program, or modify it. This freedom is protected in perpetuity because OSCAR is distributed under the terms of the GNU General Public License. The content of this User's Manual is also freely licensed under the Creative Commons License. The content of this manual applies to the latest versions of the various OSCAR applications [1].

1.2.2 About This Project

This BC billing project adds modules and enhancements to the current OSCAR BC billing system, using a component based approach. Here, we shield our volunteers from the vast and extensive

[1] `http://www.oscarcanada.org/`

OSCAR code-base by defining the entry and exit point for each component then developing it as a black-box that satisfies the requirements [2].

1.2.3 Project Input and Output

The input of this project is the XML Schema of the billing rules and advisories. The schema is provided as a XSD file to this project. This schema is shown in figure 1.1.

The output is the web application that can be added to the current OSCAR source code to let the OSCAR users write billing rules and advisories.

1.2.4 Project Deliverables

The project deliverables are:

1. **Source Code**: All developed source code for OSCAR billing project.
2. **Software Documentation**: This document (version 1.0).
3. **Installation Guide**: Installation guide for current OSCAR users.
4. **User Manual**: User manual for OSCAR users.

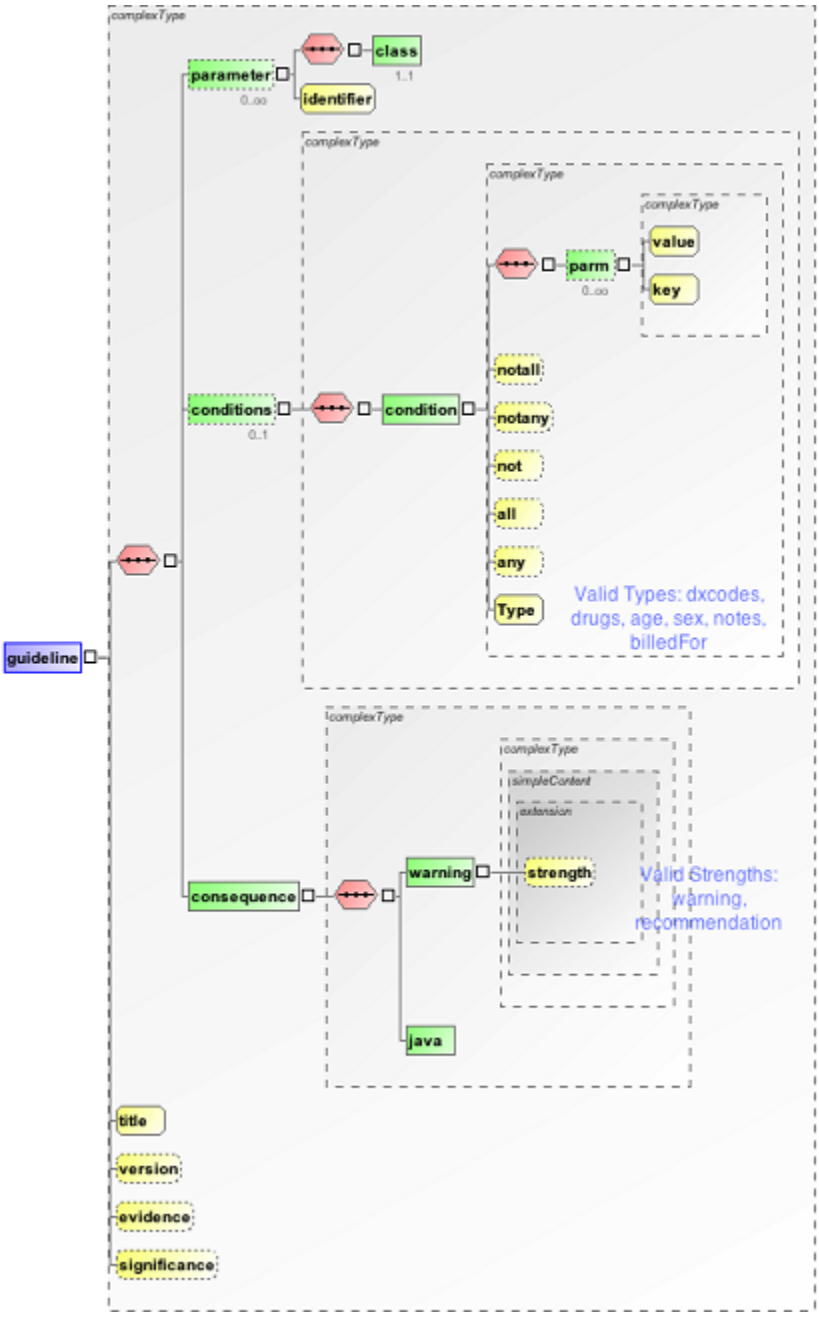

Figure 1.1: BC Billing Schema

Chapter 2

Automatic Form Generation and XForms

In this chapter, we describe our approach to automatic form generation from XML schemas using XForms. There is an introduction section 2.1. XForms are explained in section 2.2 and the automatic process is explained in section 2.3.

2.1 Introduction

Since its introduction in 1996, many real applications have migrated to use XML as a markup language since its (draft) introduction in 1996 [1].

The popularity of XML motivates us to look for tools available that automate our tasks in the project. To be precise, it will be great if there is any open source software available that receives as input the XML schema (and probably some configurations / preferences) and generates web-based forms from that schema. This way, further changes to the schema will not affect the billing component because everything will be re-generated again.

There are some commercial solutions that claim they do the job great, but we did not have the opportunity to test them. There are also a couple of open source solutions that we will shortly cover them later in this chapter. But our conclusion is, at the moment, these tools are not of great help. They still require further case based customization and the immediate output from the tools is not so good to be used directly in a widely used application like OSCAR. Still, we believe they are in the correct direction, and so, there is a lot of hope that, maybe not far from now, there will be great open source solutions that automatically generate reasonably good user interfaces from XML schemas. So, we decided to include this chapter in this document: for further reference.

2.2 XForms

In this section, we briefly introduce XForms and skim through the implementations and tools available.

[1] See `http://xml.coverpages.org/xmlApplications.html` for a list of applications

2.2.1 What is XForms?

XForms is an XML application that represents the next generation of forms for the web. XForms is not a free-standing document type, but is intended to be integrated into other markup languages, such as XHTML, ODF or SVG. An XForms-based web form gathers and processes XML data using an architecture that separates presentation, purpose and content [3].

XForms specification has not become a standard yet, it is still a recommendation from W3C.

There is also a good introduction to XForms for those interested is available at IBM web site:

`http://www.ibm.com/developerworks/xml/library/x-xformsintro1/`

2.2.2 Browser Support

Unfortunately, none of the widely used browsers (Mozilla Firefox, Internet Explorer and Google Chrome) support XForms natively.

But there is an open source project called XSLTForms[2] that allows all major browsers to manipulate XForms. XSLTForms is a client-side implementation of XForms. It is an active project and its latest release was in 2010.

Mozilla also hosts a project called "Mozilla XForms Project"[3], that provides XForms support for firefox as an extension. Mozilla XForms Project is also an active project and its last release is only a few months old.

2.2.3 Implementations

In this part, we briefly list the major implementations of XForms. For current a list of all implementations that are reported to W3C, please see their web site:

`http://www.w3.org/MarkUp/Forms/wiki/XForms_Implementations`

1. **IBM Forms**[4]: IBM Forms is a suite of products by IBM's Lotus Software division that interact to develop and deliver data-driven, XML-based electronic forms (e-forms) to end-users. IBM Forms consists of a server, designer, and client viewer that enable creation, deployment, and streamlining of forms-based processes. IBM Forms originally used Extensible Forms Description Language (XFDL) as the format for its electronic forms and it has gradually added XForms to XFDL as that standard has matured [4].

   IBM Forms is a commercial product and is neither open source nor available for free.

2. **Chiba**[5]: Chiba is an open source java implementation of XForms. It combines AJAX, XForms and Dojo toolkit to make rapid web development possible. Based on their user manual[6] it can be used when XML is used heavily in the backend to quickly generate needed user interfaces. Chiba is a server-side implementation and hence, does not require any special browser support.

   Chiba development is discontinued since 2009 when it's developers moved to betterForm.

[2] See `http://www.agencexml.com/xsltforms`
[3] See `http://www.mozilla.org/projects/xforms/`
[4] See `http://www-01.ibm.com/software/lotus/products/forms/`
[5] See `http://chiba.sourceforge.net/`
[6] See `http://chiba.sourceforge.net/ChibaUserGuide.pdf`

3. **betterForm**[7]: BetterForm is another open source implementation of XForms. It continues the work done at Chiba project. Based on their web site, betterForm allows easy creation of highly dynamic Web 2.0 user interfaces with attractive controls and layout.

   Like Chiba, betterForm is a server side implementation of XForms.

4. **X-Smiles**[8]: X-Smiles is a open source java XML browser. It supports XForms 1.0. X-Smiles is discontinued (last release was 2008).

5. **IBM XML Forms Generator**[9]: This is an eclipse plug-in that takes a WSDL[10] or an XML schema and produces XHTML with XForms. The view the form, a browser with XForms support is needed. Alternatively, other tools must be used to convert the XHTML to HTML and java script.

   IBM XML Forms Generator seems to be discontinued, or at least frozen. Its last update was more than two years ago. It requires eclipse 3.3 and there are some compatibility issues with newer versions.

2.3 Automatic XForm Generation

2.3.1 Client Side Implementations

Client side implementations (like XSLTForms) are not of our interest due to security issues. A client side verification can easily be bypassed by a clever! client, so, server side validation is still required. The Open Web Application Security Project (OWASP)[11] also recommends not relying on client side data validation. See their cite for further information:

`http://www.owasp.org/index.php?title=Reviewing_Code_for_Data_Validation&setlang=es#Never_Rely_on_Client-Side_Data_Validation`

2.3.2 WSDL2XForms and betterForm

BetterForm is the most active server-side implementation of XForms. To be able to use it, we must first convert our XML schema to XForms. This is done using WSDL2XForms [12]. WSDL2XForms is an open source project implemented with java that generates XForms from given WSDL or XSD files.

We used the latest version of WSDL2XForms[13] available for download. This version is bundled with useful scripts (for both Linux and Windows environments) that makes it more easy to use. To convert the billing schema to XForms, the following command can be executed (in windows environment):

```
schema2xforms.bat [path-to]bcbilling.xsd [path-to-new]bcbilling.xls "element=editix:guideline"
```

Installing and running betterForm is quite straight-forward:

1. Download installer jar file from betterForm source-forge page: `http://sourceforge.net/projects/betterform/files/limeGreen\%20RC1/betterFORM-install.jar/download`

[7] See `http://www.betterform.de/`
[8] See `http://www.xsmiles.org/xsmiles_objectives.html`
[9] See `http://alphaworks.ibm.com/tech/xfg/download`
[10] Web Service Definition Language
[11] See `http://www.owasp.org`
[12] See `http://sourceforge.net/projects/wsdl2xforms/`
[13] wsdl2xforms-20100118.tar.gz

2. Run the jar file to start installation: `java-jarbetterForm-install.jar`
3. Follow the installation steps
4. Start betterForm from `[Installpath]/startup.bat`
5. Point your browser to your local site `http://localhost:8080/betterform`
   BetterForm dashboard is shown in figure 2.1.

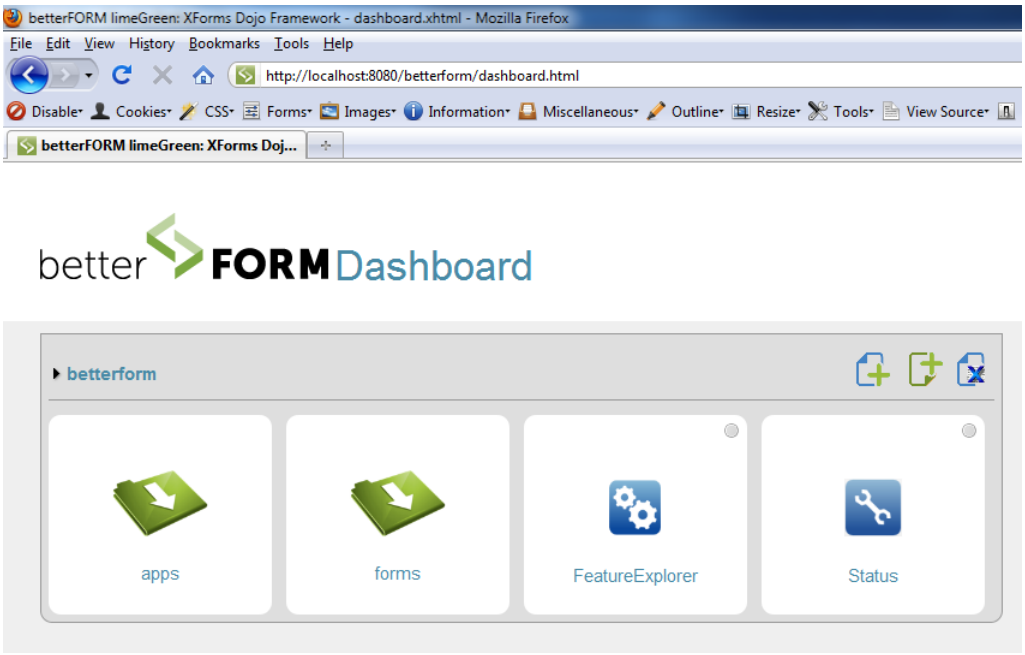

Figure 2.1: BetterForm Dashboard

6. Open "Forms" and click on "Upload your form" button (as shown in figure 2.2).
7. Open your form! (See figure 2.3).

The first impression is that "the result is great", but unfortunately, this impression does not last long. Just start playing with the form and you will soon find that "Delete"s do not work!

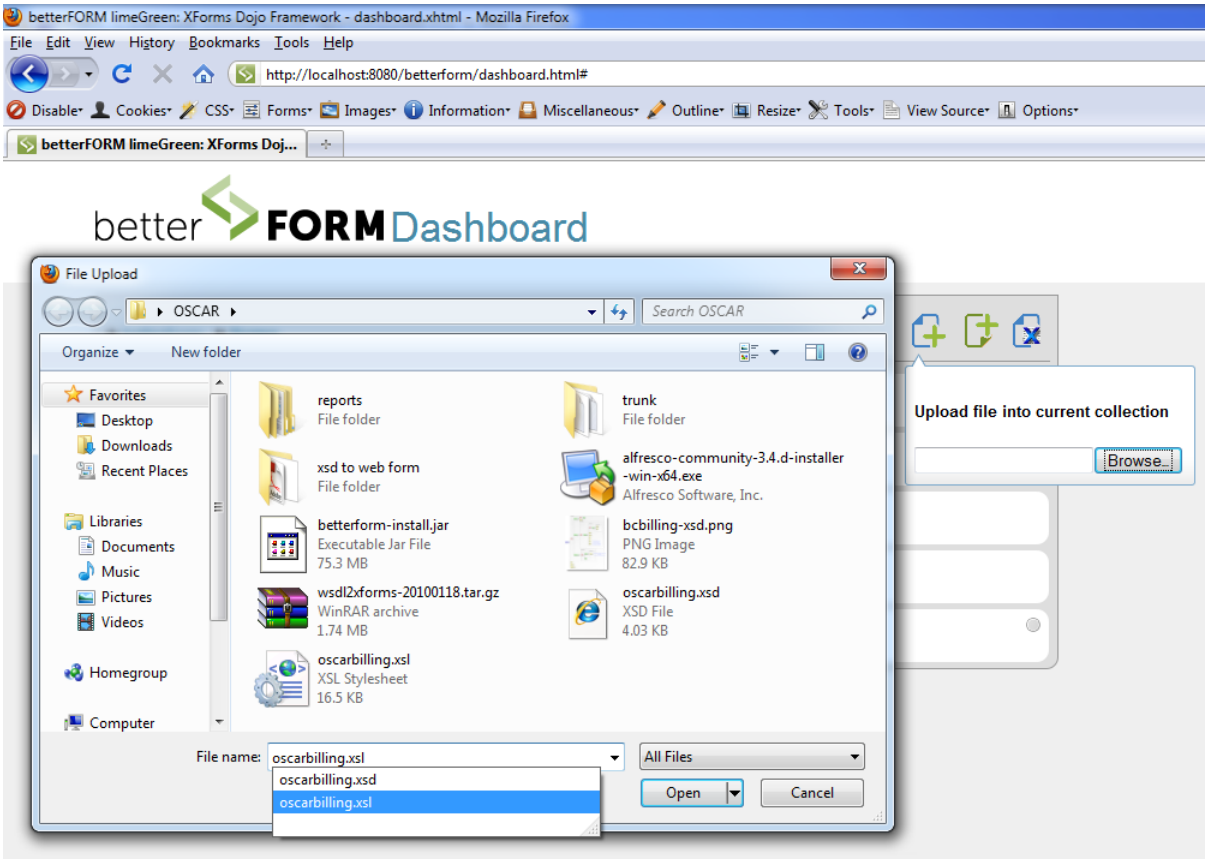

Figure 2.2: BetterForm Upload Form (XSL file)

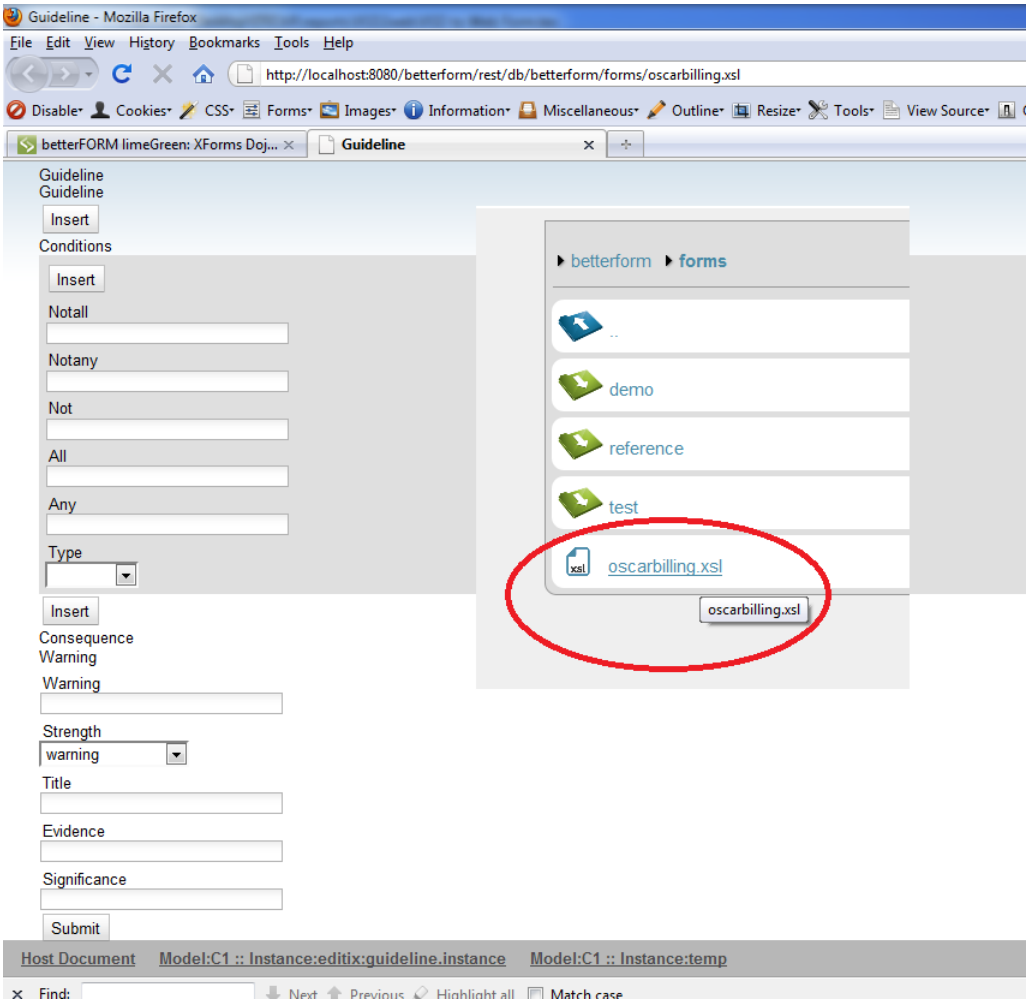

Figure 2.3: BetterForm Final Output

Chapter 3

BC Billing Architecture

In this chapter, the architecture of the BC Billing web site is explained. General design of the system is presented in section 3.1 and an overview of the components is described in 3.2.

3.1 General Architecture

3.1.1 Technology

We have selected Java Servlet Technology for BC Billing website project. Java Servlet technology provides Web developers with a simple, consistent mechanism for extending the functionality of a Web server and for accessing existing business systems [5].

The key reasons why Java Servlet Technology is chosen for this project includes (but is not limited to) the following:

- It does not require installing any specific software on the clients machines (including Java), because it is entirely run in server-side. The only requirement of the client machine is that it must be able to run any if the recent web browsers. A servlet can almost be thought of as an applet that runs on the server side [5].
- With Java Servlet Technolog, we have the most compatibility possible with the existing OSCAR software.

3.1.2 General Design

The BC Billing project is designed based on the MVC (model-view-controller) architecture:

- **User Interface**: User interface is the *VIEW* of the project. It provides the functionality of the system in a human readable / usable format.
- **Servlet**: The servlet is the *CONTROLLER*. It is the heart of the project that connects the view and the model.
- **Billing Schema Library**: Schema library defines the *MODEL*. It implements / provides the API for the BC Billing schema.

The general design of the project is shown in figure 3.1.

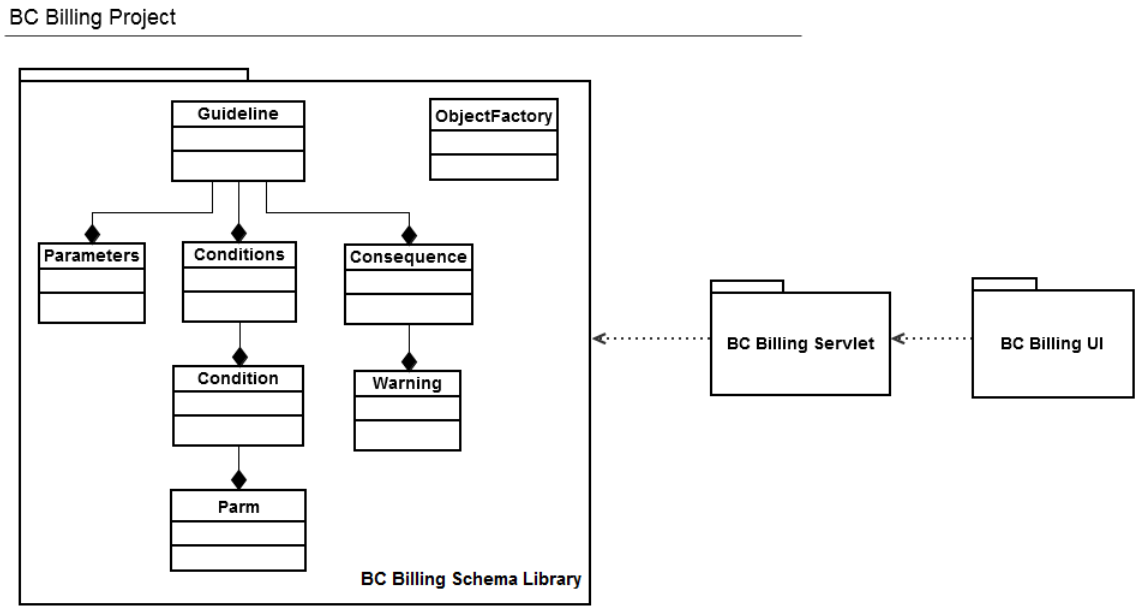

Figure 3.1: BC Billing User Interface

3.2 Components Overview

3.2.1 User Interface Component

BC Billing user interface is the view of the system. It is a dynamic web form written in HTML and java script. This component is responsible for the following:

- Receiving billing data from user and sending it to the billing servlet
- Receiving the XML generated from billing data and provide it to the user

BC Billing user interface is shown in figure 3.2.

3.2.2 Servlet Component

BC Billing servlet is the controller of the project. It is responsible for responsing the user requests sent from the user interface. In the BC Billing project, the only request sent from the user interface is the request of generating the XML file from the billing information (which is provided by the user). It implements the standard java servlet interface HttpServlet and uses the billing schema library classes for XML generation.

3.2.3 Billing Schema Library

BC Billing schema library is the model of the project. It implements a library of helper classes required to generate billing XML files based on the billing XML schema. It consists of the following classes:

- **ObjectFactory**: Java class that allows us to programmatically construct new instances of the java representation for XML content.
- **Guideline**: Java class generated for *guideline* tag in BC Billing Schema. Guideline is the root tag.
- **Guideline.Consequence**: Java class generated for *consequence* tag in BC Billing Schema.
- **Guideline.Consequence.Warning**: Java class generated for *warning* tag in BC Billing Schema.
- **Guideline.Conditions**: Java class generated for *condition* tag in BC Billing Schema.
- **Guideline.Conditions.Condition**: Java class generated for *condition* tag in BC Billing Schema.
- **Guideline.Conditions.Condition.Parm**: Java class generated for *parm* tag in BC Billing Schema.
- **Guideline.Parameter**: Java class generated for *parameter* tag in BC Billing Schema.

These classes are all automatically generated using JAXB[1]. JAXB stands for Java Architecture for XML Binding. JAXB enables java programmers generate java source code from their XSD data definition.

[1] See `http://jaxb.java.net`

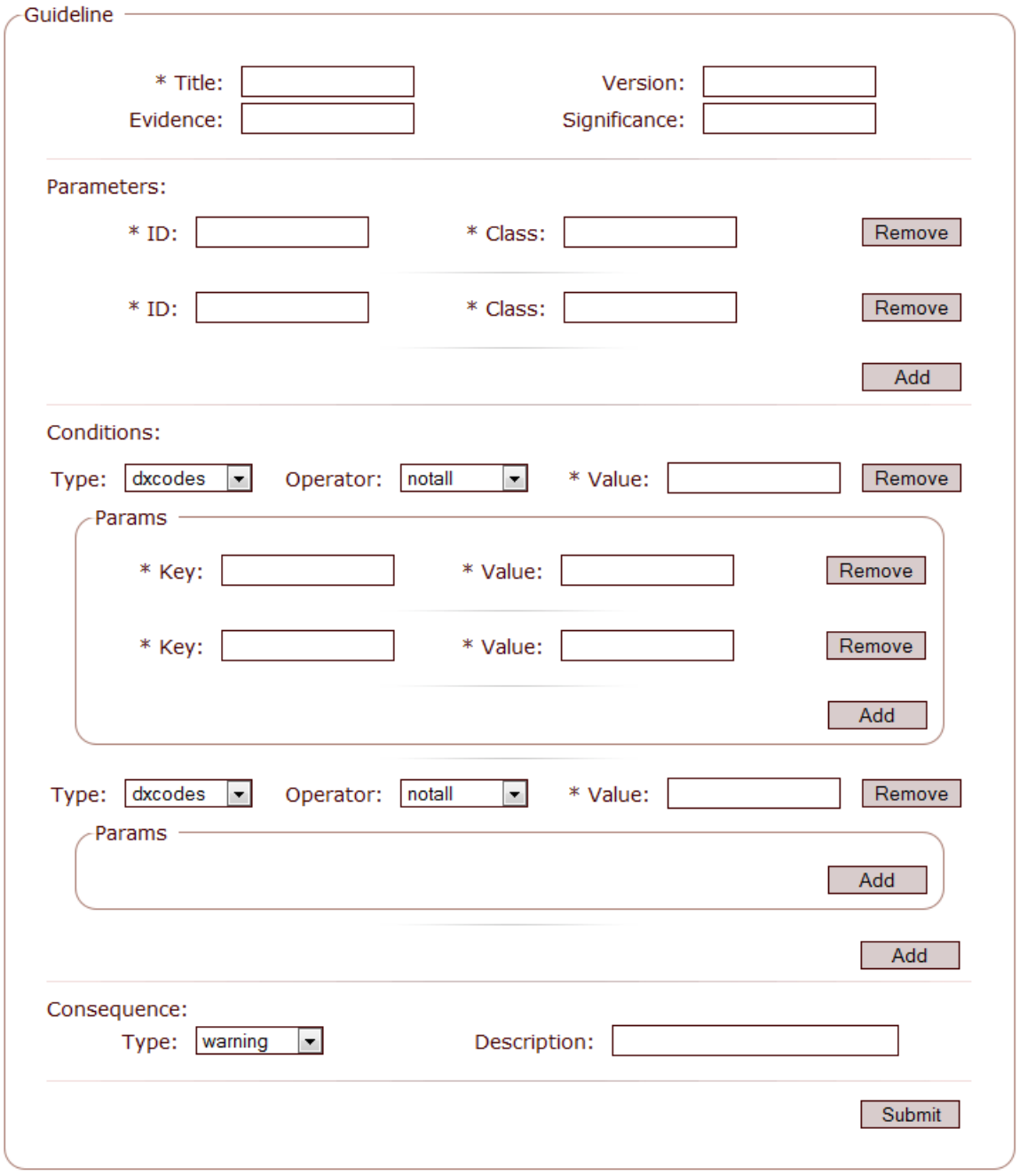

Figure 3.2: BC Billing User Interface

Bibliography

[1] OSCAR User's Manual Web site, http://oscarmanual.org/.

[2] OSCAR User's Manual Web site, http://oscarbcbilling.patesco.ca/.

[3] XForms 1.1 W3C Recommnedation, http://www.w3.org/TR/xforms/.

[4] IBM Lotus Forms, Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/IBM_Lotus_Forms.

[5] Java Servlet Technology, http://www.oracle.com/technetwork/java/javaee/servlet/index.html.