Implementation of XSLT-based Schema Mapper using RCP

Tae-Jin Ha*, Hye-Ja Bang
Dept of Computer Science & Engineering, Seoul National University of Technology

Summary
RCP(Rich Client Platform) is a Java development platform to develop Rich Client based on OSGi framework. This paper proposes a method of designing Schema Mapper by using RCP based on XSLT(eXtensible Stylesheet Language Transformation) of W3C(World Wide Web). The Schema Mapper defines each document structure of the Source that offers data and the Target that handles data as schema and makes XSLT document which is defining translation regulation between two schemas. The created XSLT document shows original schema, Translation regulation, and purpose schema in order on the editor program as pattern of tree. The Schema Mapper implemented in this paper under WYSIWYG(What You See Is What You Get) can write, edit, and test translation regulation and offers special edit program for a variety of document format and schema document such like XML(eXtensible Markup Language), DBF(Data Base Format), and UDF(Usesr Defined Format). It also designs and implements flexibly to be able to add new document format easily.

Key words:
XSLT, Mapper, RCP

1. Introduction

The Java development environment for today's GUI(graphic user interface) is approaching a new phase because of RCP. The RCP is a Java development-tool platform to develop Rich client framework based on OSGi framework [1]. Currently in company's server, translation engine and translation regulation have problems with distribution, expansion, and maintenance. It is impossible for them to add new document format. The necessity of translation engine has been increasing to complement these design problems, to provide document format of existing XML[2], DBF and UDF, and to supply new document format such as EDI.
However, it is essential to have Rich Client Development-Tool Platform based on Open API to develop efficiently the editor of XSLT document, which defines schema document and translation regulation as XSL, with short time and a little bit of money before developing a new translation engine.
This paper implements Schema Mapper by using RCP based on XSLT which is document-translation standard of W3C.
In this paper, the implemented Schema Mapper can edit XML, DBF, UDF document, XSD(XML Schema Definition), DTD(Document Type Definition), USD(UDF Schema Definition), XSL, and translation file(XT) by using the editor and offer WYSIWIG environment which is able to check the stage of translation. Also, this paper proposes the method of constructing the Rich Client System which is able to distribute, expand, and reuse.

Manuscript received May 5, 2009
Manuscript revised May 20, 2009

2. RCP (Rich Client Platform)

RCP is a Java development-Tool platform to develop Rich Client based on OSGi framework. The platform consists of largely runtime, SWT / JFace, and UI workbench

2.1 OSGi Framework

Plug-in component model of RCP is based on implementation of OSGi framework 4.0 statement (http://osgi.org). OSGi statement constructs a framework to define, combine, and run component or bundle. The bundle is considered as implementation of plug-in. The word, called plug-in, has been used to mean component.

2.2 Runtime

Runtime includes major mechanism such as application model, extension registry, and so on. Application is a running program such as main() method of java program in the runtime. Product is a higher concept than application. If application is defined as product, it can offer environment like Branding, Splash, and Launching. The Schema Mapper implemented in this paper is defined as such products.

2.3 SWT / JFace

SWT is placed below runtime and low-level graphic library based on window system. SWT is light and fast because it uses a great many native widget and offers lookandfeel like window system.
JFace offers packages of component related to image, font registry, text, dialog window, and environment setting. Schema Mapper implemented in this paper expands and implements perspective, view, editor, action, and dialog of RCP so it can reduce time and effort to make new component.

2.4 UI workbench

Workbench offers contribution-based UI expansion and adds window, perspective, view, editor, action, and so on into workspace. Contribution offers expansion that can define UI components explicitly without coding. This function of expansion allows Schema Mapper to add components such as action, command, and magic tool.

3. XML Schema Mapping

XML Schema includes not only structure, but also each element name, explanation, type, and restriction of documents necessary to make final XML instance document.

3.1 Types of Mapping

Mapping forms relations between the elements of the same meaning from different schema structure. And that indicates relation between simple elements and can also indicate relations which deal with extra process.
Mapping types are divided into mapping between the documents and mapping between the nodes in the document. The correspondence between documents forms three types such as 1:1, 1:N, and M:N depending on the number of original and target documents[4]. In this paper, Schema Mapper is type of 1:1 mapping because it decides correspondence between documents in the translator.
The next is correspondence type between the nodes in document. The node in document becomes element and attribute of schema and forms four types such as 1:1, 1:N, M:1, and M:N like correspondence between documents. The correspondence between these nodes needs functions, that is, in 1:N case, it needs the function(substr) which is able to divide a sentence and, in M:1 case, it needs the function(concat) that combines words.

3.2 XSLT

XSLT plays the role in transforming or remaking XML document into different document form as document translation standard languages of W3C. The usage type of XMLT templates can be divided into following:

- One template: one template can have every translation regulations. It translates documents in the order of visits to Original XML document and makes progress in top-bottom way.
- Applying by mode: if it decides mode when a template is declared, it is possible to modulate through the way calling as <apply-template>
- Call by name: when template is declared, it decides original name and calls <call-template>. It is not only possible to modulate, but also easy to control the transform order of Target schema.

3.3 Translation File(XT)

Translation file inmplemented in this paper based on XSLT saves the types and locations of original schema and target schema, and mapping XSL into one file as XML form. In XSD type case, it selects root node and saves when translation file is created. Because Schema Mapper translation makes progress based on XSL, it is implemented to offer the functions of exporting XSL and importing XSL.

3.4 Translation Engine

The results of comparison between the existing translation engine and the implemented Schema Mapper are displayed in Table 1. The big difference is that implemented Schema Mapper is able to add new format and offering exclusive schema editor during using schema simultaneously

Table 1. Comparison with the existing translation engine

	BW Mapper	GeT*Mate	Xmap	Schema Mapper
Available OS	ALL (Java)	ALL (Java)	ALL (Java)	ALL (Java)
Support Transformation Format	EDIFACT XML	XML	XML UDF(Flat File)	XML, DBF UDF(Flat File)
Script Type	Xquery	XSLT	XSLT	XSLT
Transformation Method	XML: Schema / Xquery EDI: Data Dictionary	XML: XSLT	XML: XSLT	XML: Schema / XSLT
Maintenance	Impossible	Possible	Impossible	Possible
If a new document, needed	Data Dictionary Xquery	XSLT	UDF : Rule XSLT	UDF : Rule XSLT
Add a new format	Impossible	Impossible	Impossible	Possible
Export type	Install	Install	Library	Library / Product
Product dependency	BusinessWare	None	None	None
Schema Editor	None	None	None	All

4. Design and implementation of Schema Mapper

4.1 Total structure of Schema Mapper

Schema Mapper is largely composed of translation engine and Mapper GUI. The translation engine consists of Schema model and input-output device, and translation device. The Mapper GUI consists of the workbench such as schema editor, translation file editor, and so on. Figure 1 shows the structure of Schema Mapper.

4.2 Schema Data Model

Schema data model is divided into DTD, XSD, and USD data model. The DTD uses dtdparser schema model of wutka.com. XSD data model are composed of Document, Element, and Attribute. USD data model consists of Document, Group, and item.

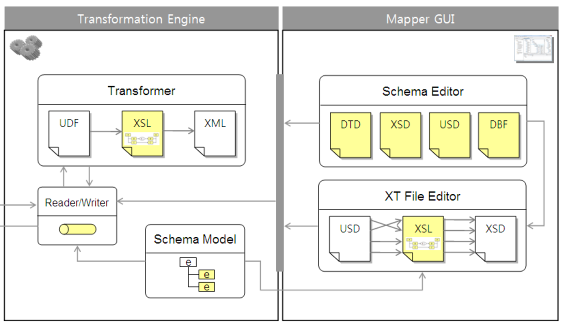


Figure 1. Structure of Schema Mapper

4.3 Input/Output

For Input/Output implemented original-file Reader and target-file Writer are implemented as interface. Because translation refers to Reader, Writer and interface, it is possible to expand new document format additionally if Input/Output class is implemented about file and schema by inheriting reader and writer interface.
Reader is divided into original file, original schema, and Reader. The original file input includes XML and UDF. And original schema input is divided into DTD, XSD, USD schema input. Translation file input consists of XSL and XT document.
Writer is divided into target file, target schema, and writer printing translation file. The target file output is divided into the form of XML and UDF. However, the target schema output implements only USD schema output. Therefore, the DTD and XSD schema file's output uses exclusive plug-in of RCP.

4.3 Translator

Translator creates DOM entity by examining XML, DBF, and UDF document and each schemas or document information of XSLT. And, the translator analyses the creation of structure and presentation information by using DOM entity. Finally, the Translator checks fundamental validation. Figure2 shows the structure of translator and flow of document of UDF->XML transformation.

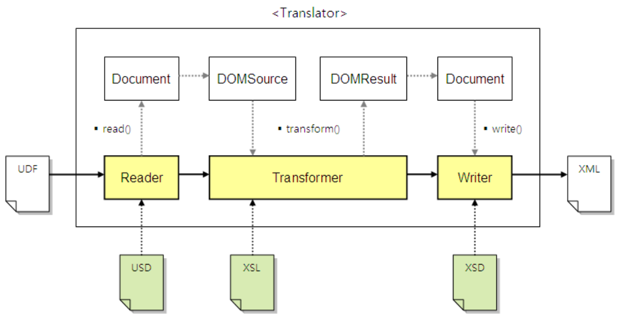
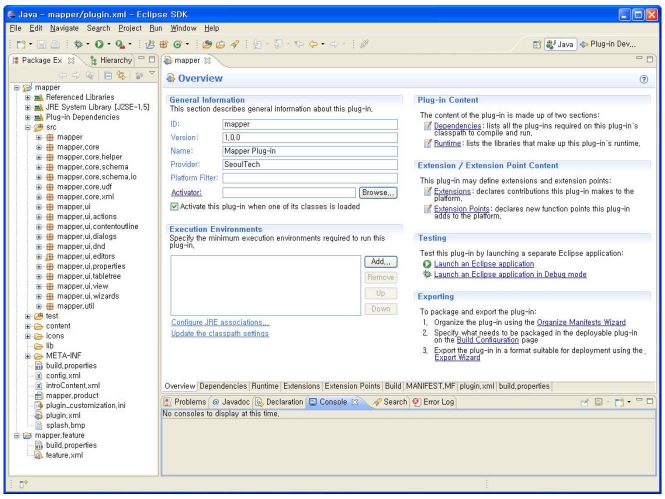


Figure 2. Translator Structure

4.4 Workbench implementation

Implementation environment of Schema Mapper is IBM-PC compatible computer (Pentium M, 1.7G). And operating system is Windows XP Home Edition and Service Pack 3. Also, Java 1.5 and eclipse RCP 3.3.2 were used. The library of xerces, xalan, dtdparser, and dom4j was used to verify and analyze document. Figure 3 shows development environment to implement Schema Mapper.


Java - mapper/plugin.xml - Eclipse SDK

File Edit Navigate Search Project Run Window Help

Package Ex | Hierarchy

```
mapper
  Referenced Libraries
  JRE System Library [J2SE-1.5]
  Plug-in Dependencies
  src
    mapper
    mapper.core
    mapper.core.helper
    mapper.core.schema
    mapper.core.schema.io
    mapper.core.udf
    mapper.core.xml
    mapper.ui
    mapper.ui.actions
    mapper.ui.contentoutline
    mapper.ui.dialogs
    mapper.ui.dnd
    mapper.ui.editors
    mapper.ui.properties
    mapper.ui.tabletree
    mapper.ui.view
    mapper.ui.wizards
    mapper.util
  test
  content
  icons
  lib
  META-INF
  build.properties
  config.xml
  introContent.xml
  mapper.product
  plugin_customization.ini
  plugin.xml
  splash.bmp
mapper.feature
  build.properties
  feature.xml
```

mapper

Overview

General Information
This section describes general information about this plug-in.

ID:	mapper
Version:	1.0.0
Name:	Mapper Plug-in
Provider:	SeoulTech
Platform Filter:	
Activator:	

☑ Activate this plug-in when one of its classes is loaded

Execution Environments
Specify the minimum execution environments required to run this plug-in.

Configure JRE associations...
Update the classpath settings

Plug-in Content
The content of the plug-in is made up of two sections:
Dependencies: lists all the plug-ins required on this plug-in's classpath to compile and run.
Runtime: lists the libraries that make up this plug-in's runtime.

Extension / Extension Point Content
This plug-in may define extensions and extension points:
Extensions: declares contributions this plug-in makes to the platform.
Extension Points: declares new function points this plug-in adds to the platform.

Testing
Test this plug-in by launching a separate Eclipse application:
Launch an Eclipse application
Launch an Eclipse application in Debug mode

Exporting
To package and export the plug-in:
1. Organize the plug-in using the Organize Manifests Wizard
2. Specify what needs to be packaged in the deployable plug-in on the Build Configuration page
3. Export the plug-in in a format suitable for deployment using the Export Wizard

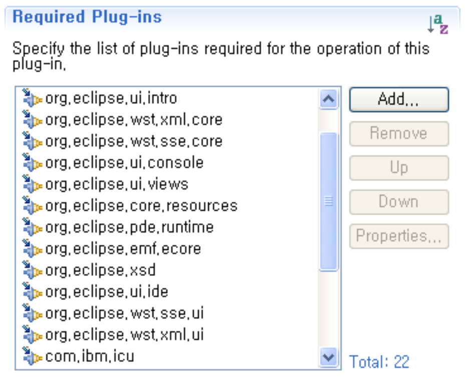
Overview | Dependencies | Runtime | Extensions | Extension Points | Build | MANIFEST.MF | plugin.xml | build.properties

Problems | Javadoc | Declaration | Console | Search | Error Log

No consoles to display at this time.

Figure 3. Development Environment of Schema Mapper(RCP)

Schema Mapper registers help tool, console, resource navigator, DTD, and XSD schema Editor as Plug-in. Figure 4 is the Plug-in configuration of Schema Mapper .


Required Plug-ins

Specify the list of plug-ins required for the operation of this plug-in.

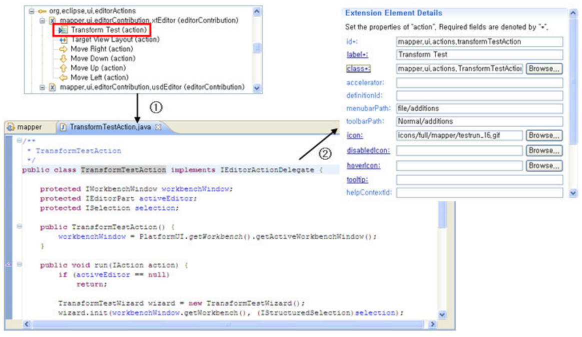
```
org.eclipse.ui.intro
org.eclipse.wst.xml.core
org.eclipse.wst.sse.core
org.eclipse.ui.console
org.eclipse.ui.views
org.eclipse.core.resources
org.eclipse.pde.runtime
org.eclipse.emf.ecore
org.eclipse.xsd
org.eclipse.ui.ide
org.eclipse.wst.sse.ui
org.eclipse.wst.xml.ui
com.ibm.icu
```

Add... | Remove | Up | Down | Properties...

Total: 22

Figure 4. Plug-in configuration of Schema Mapper

One of useful functions of RCP is expansion. Figure 5 shows the process of adding Translation-Test action in Editor contribution, implementing action class, and configuring menu, tool bars, action location of path, and icon Image.


org.eclipse.ui.editorActions
mapper.ui.editorContribution.xtEditor (editorContribution)
Transform Test (action)
Target View Layout (action)
Move Right (action)
Move Down (action)
Move Up (action)
Move Left (action)
mapper.ui.editorContribution.usdEditor (editorContribution)
①
②
mapper
TransformTestAction.java

Extension Element Details

Set the properties of "action". Required fields are denoted by "*".

id*:	mapper.ui.actions.transformTestAction
label*:	Transform Test
class*:	mapper.ui.actions.TransformTestActio
accelerator:	
definitionId:	
menubarPath:	file/additions
toolbarPath:	Normal/additions
icon:	icons/full/mapper/testrun_16.gif
disabledIcon:	
hoverIcon:	
tooltip:	
helpContextId:	

```
/**
 * TransformTestAction
 */
public class TransformTestAction implements IEditorActionDelegate {

    protected IWorkbenchWindow workbenchWindow;
    protected IEditorPart activeEditor;
    protected ISelection selection;

    public TransformTestAction() {
        workbenchWindow = PlatformUI.getWorkbench().getActiveWorkbenchWindow();
    }

    public void run(IAction action) {
        if (activeEditor == null)
            return;

        TransformTestWizard wizard = new TransformTestWizard();
        wizard.init(workbenchWindow.getWorkbench(), (IStructuredSelection) selection);
```

Figure 5. Process of adding action by using Extension

When implementation of the Schema Mapper is finished, products are distributed. First, click the right button on the project so run distribution tool. Then, select the target folder and then press the complete button. Figure 6 shows the process of distributing the completed Schema Mapper.

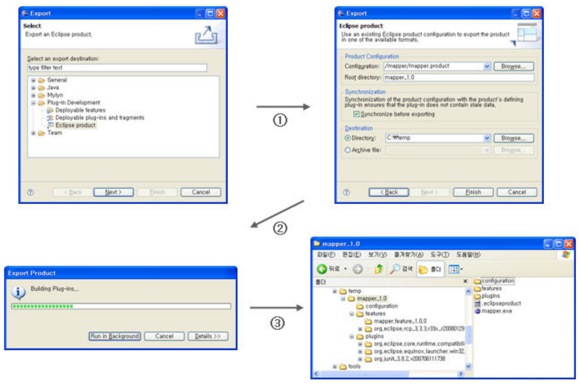


Figure 6. Export process of Schema Mapper

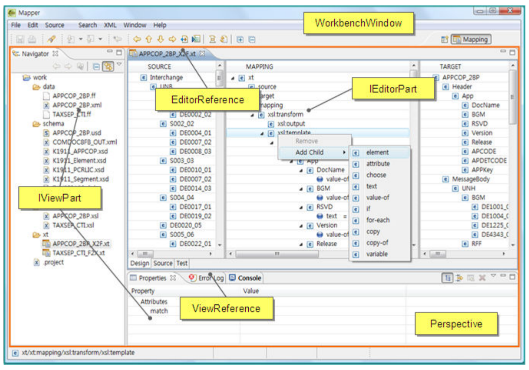
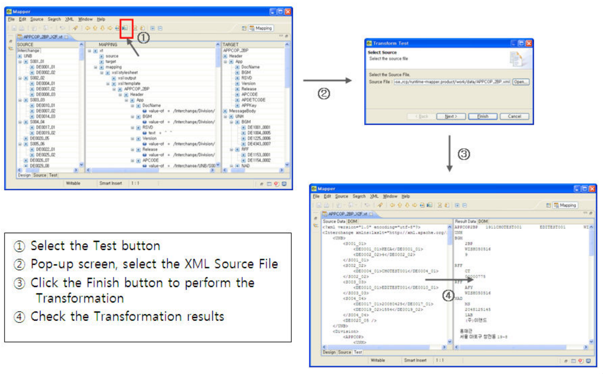


Figure 7. Workbench of Schema Mapper

Figure 7 shows workbench performance screen of the completed Schema Mapper. The navigator on the left shows the file list of working folder. File editor is on the top center of the screen. The original Schema tree is located on the left, translation regulation is located at the center, and the target schema tree is located on the right of File translation Editor. Property, error, and console screen are located at the bottom.

After Translation File is completed, Figure 8 shows the process of performing the translation test to check mapping errors. After performing translation by clicking the translation test button, compare the data between original file and target file



① Select the Test button
② Pop-up screen, select the XML Source File
③ Click the Finish button to perform the Transformation
④ Check the Transformation results

Figure 8. Process of translation test

5. Conclusion

This paper designs and implements Schema Mapper offering a variety of document format schema such as XML, DBF, and UDF and translation regulations between those schemas under WYSIWYG environment based on XSLT which is available for writing and editing efficiently by using RCP, Rich Client develop-tool platform based on OSGi framework. Furthermore, existing translation system can offer the function of translation about only limited document format and schemas and cannot add new document format. However proposed system solves those problems using flexible input-output interface and java standard XML translator. And it offers version, distribution, and update of production through a variety of Plug-In offered by RCP.

This Schema Mapper based on XSLT will replace the existing translation system so that it will offer a variety of document format and translation between the schemas and be used efficiently to edit schema document and translation regulations. Also, Rich Client development skill that is schema editor and translation regulation editor using RCP is considered to assist in activating WYSIWYG-based Schema Mapper.

Further studies need to offer not only XML, DBF, UDF schema, but also a variety of document format, to offer the translation between the schemas, and to be improved to be able to edit Original schema and Target schema directly on the edit application.

References

[1] Jeff McAffer, Jean-Michael Lemieux, “Eclipse Rich Client Platform, Designing, Coding, and Packaging Java Applications”, Addison-Wesley, 2005.
[2] World Wide Web Consortium, Extensible Markup Language (XML) 1.0 (Fifth Edition), W3C Recommendation, http://www.w3c.org/TR/xml, 2008.
[3] World Wide Web Consortium, XSL Transformations (XSLT) Version 2.0, W3C Recommendation, http://www.w3c.org/TR/xslt20, 2007.
[4] Cheol-Min Shin, Kyong-Ha Lee, Kyu-Chul Lee, “XTrans: The XML Schema Mapper considering transformation feasibility”, Paper of Korea Computer Congress Vol. 33, 2006.
[5] Mun-Chan Seo, “Develop a Schema Mapper for the transformation of XML documents”, Master's Thesis of Chungnam National University, 2006
[6] Chang-soo Kim (el), “A Study of XSLT document editing system for XML document transformation”, Korea Institute of Maritime Information and Communication Sciences Spring General Congress Vol. 8, No. 1, 2004
[7] Jong-Chul Song (el), “XML structure translation system using schema structure data mapping”, Paper of Korean Institute of Information Scientists and Engineers Vol. 10, No. 5, 2004.

Tae-Jin Ha* **(河 太 振)**
Bachelor of Engineering(Computer Science), Seoul National University of Technology.
Master Course(Computer Science), Seoul National University of Technology.
Interests: Programming languages, Software engineering

Hye-Ja Bang (方 惠 子)
Bachelor of Engineering (computer science), Soongsil University
Master (Electronic calculations), University of North Texas
Exchange professor(Computer Science), Florida State University
Current: Professor of computer science, Seoul National University of Technology
Interests: Programming language, Compiler, Formal language