RVOT: A Tool For Making Collections OAI-PMH Compliant

K. Sathish, K. Maly, M. Zubair

Computer Science Department
Old Dominion University
Norfolk, Virginia USA
{kumar_s,maly,zubair}@cs.odu.edu

X. Liu

Research Library
Los Alamos National Laboratory
Los Alamos, New Mexico USA
liu_x@lanl.gov

Abstract

A number of on-line journals and scientific digital libraries (DL) exist today, however, there is a lack of interoperability between these libraries. Researchers have been looking at this issue and one of the major efforts to address technical interoperability among distributed archives is the Open Archives Initiative (OAI). The objective of OAI is to develop a low-barrier, lightweight framework to facilitate the discovery of content in distributed archives. In this paper, we describe a tool, Rapid Visual OAI Tool (RVOT), which can help small organizations in making their collections OAI-PMH (OAI Protocol for Metadata Harvesting) compliant in a quick and convenient way. RVOT can be used to graphically construct an OAI-PMH repository from a collection of files. It includes a metadata translation tool, a lightweight HTTP server, and an OAI-PMH request handler. The records in the original collection can be in any of the supported formats including RFC1807, MARC subset, and COSATI subset formats. RVOT helps to define a mapping visually from native format to Dublin Core (DC) format, once the mapping is defined, native metadata files are converted to DC format metadata files, and the repository becomes OAI-PMH compliant. RVOT is self-contained, easy to install, and can be extended to support new metadata formats.

1 Introduction

A number of on-line journals and scientific digital libraries exist today, however, there is a lack of interoperability between these libraries. One of the major efforts to address technical interoperability among distributed archives is the Open Archives Initiative (OAI) [5,9]. The objective of OAI is to develop a low-barrier, lightweight framework to facilitate the discovery of content in distributed archives. The OAI framework supports data providers (repositories, archives) and service providers. Service providers develop value-added services based on the information collected from data providers. Data providers are simply collections of harvestable metadata that may or may not contain additional services and content.

The digital library community is building a variety of tools that can help in building, evaluating and testing of OAI-PMH compliant libraries [6]. In Table 1 we categorize OAI-PMH related tools. On the data provider side, application packages (e.g. Dspace [11]) provide complete software for building a data provider with an integrated OAI-PMH module. Similarly, on the service provider side we have off-the-shelf application packages (e.g. CDSware [1]) and development frameworks. These packages can be used either to build a digital library from scratch or customize an existing digital library.

An existing digital library can be made OAI-PMH compliant by adding a software layer, which receives OAI-PMH requests over HTTP, interacts with the digital library to fetch the requested information and sends out the results. Typically, this interaction involves fetching the required metadata from the underlying storage scheme followed by mapping of the metadata to mandatory Dublin Core (DC) [14] format required by OAI-PMH before sending it out. As this software layer needs to understand the underlying DL implementation, it tends to become specific for a DL. For large organizations, development of such a software layer specific to their library is not a major investment. For this purpose one can possibly use development frameworks such as NCSA Cocoa [4]. However, small organizations would prefer a common tool that can be configured to make their small collection OAI-PMH compliant. For creating a new OAI-PMH library from scratch, one can use application packages like EPrint from Southampton [2]. However, the EPrint software because of its installation and maintenance complexity is suitable mostly for large organizations.

In this paper, we describe a tool, Rapid Visual OAI Tool (RVOT), which can help small organizations in making their collections OAI-PMH compliant. The RVOT is designed to make small or medium repositories OAI-PMH compliant quickly and

Table 1. OAI-PMH Related Tools

Category		*Tools*
Data Provider	Application with OAI component	Dspace, eprints.org, Kepler
	Development framework	UIUC OAI Implementation, OCLC OAICat, VTOAI package, oaiperl, NCSA Cocoa
Service Provider	Application with OAI component	Arc, CDSware, Clelestial, DP9, Repository Explorer
	Development framework	OCLC OAIHarvester, oaiperl, my.OAI

conveniently with very limited cost. RVOT can be used graphically to construct an OAI-PMH repository from a collection of files. It includes a metadata translation tool, and a lightweight HTTP server including an OAI-PMH request handler. The records in the original collection can be in any of the supported formats including RFC1807 [7], MARC [3] subset, and COSATI subset formats. RVOT helps visually to define a mapping from native format to DC format; once the mapping is defined, native files are converted to DC files, and the repository becomes OAI-PMH compliant. The design of RVOT is such that it can be extended to support new metadata formats. The tool is self-contained; it comes with a lightweight HTTP server and an OAI-PMH handler. The source code of RVOT is available at *http://rvot.sourceforge.net/*.

2 Metadata Conversion and OAI-PMH Request Handling

To make a repository OAI-PMH compliant, we need to build support for (a) translating native metadata into a standard metadata format required by OAI-PMH, and (b) handling OAI-PMH requests. The RVOT provides the capability to convert native metadata to DC format and the ability to handle OAI-PMH requests for metadata. We now describe the processes that have been implemented in RVOT for handling metadata conversion and OAI-PMH requests, and then describe how the embedded OAI webserver handles OAI-PMH requests.

2.1 Metadata Conversion

The metadata conversion process is illustrated in Figure 1. The conversion of native metadata files to DC begins by the user[1] first selecting the native format, specifying the location of native metadata files and the location for storing the translated DC files. Once the locations are specified, the metadata parser corresponding to the selected native format is invoked to parse the metadata files and extract the set of elements used across all the metadata files. Using the metadata-mapping interface the user specifies a mapping between native elements and DC elements. Once the mapping is specified, the metadata parser is again invoked to perform the conversion. The converted files are stored in the location specified for DC files. A more detailed explanation is provided in Section 3 and a sample metadata conversion process is described in Section 5.

2.2 Handling OAI-PMH Request

RVOT comes with an OAI-restricted webserver (it does not handle any other HTTP requests), which has two components: HTTP server and OAI Handler. The HTTP server receives OAI-PMH request embedded in an HTTP request. The request is passed to the OAI-PMH handler. The handler parses the request to get the OAI-PMH verb that indicates the type of requests [8]. Based on the verb, the OAI-PMH handler generates the response using the metadata available in its repository. A more detailed explanation is provided in Section 3. Figure 1 shows the flow of requests and responses to the OAI webserver.

3 The Rapid Visual OAI Tool Architecture

The RVOT application package consists of the following major modules.

- Metadata Manager
- OAI Webserver
- Integrated Graphical User Interface (GUI).

Figure 2 illustrates the architecture of RVOT and the interactions of the various components.

3.1 Metadata Conversion

Logically this module can be further classified into three sub-modules

- Native to DC Mapping Definition Tool.
- Native to DC Metadata Converter.
- Metadata Publishing Tool.

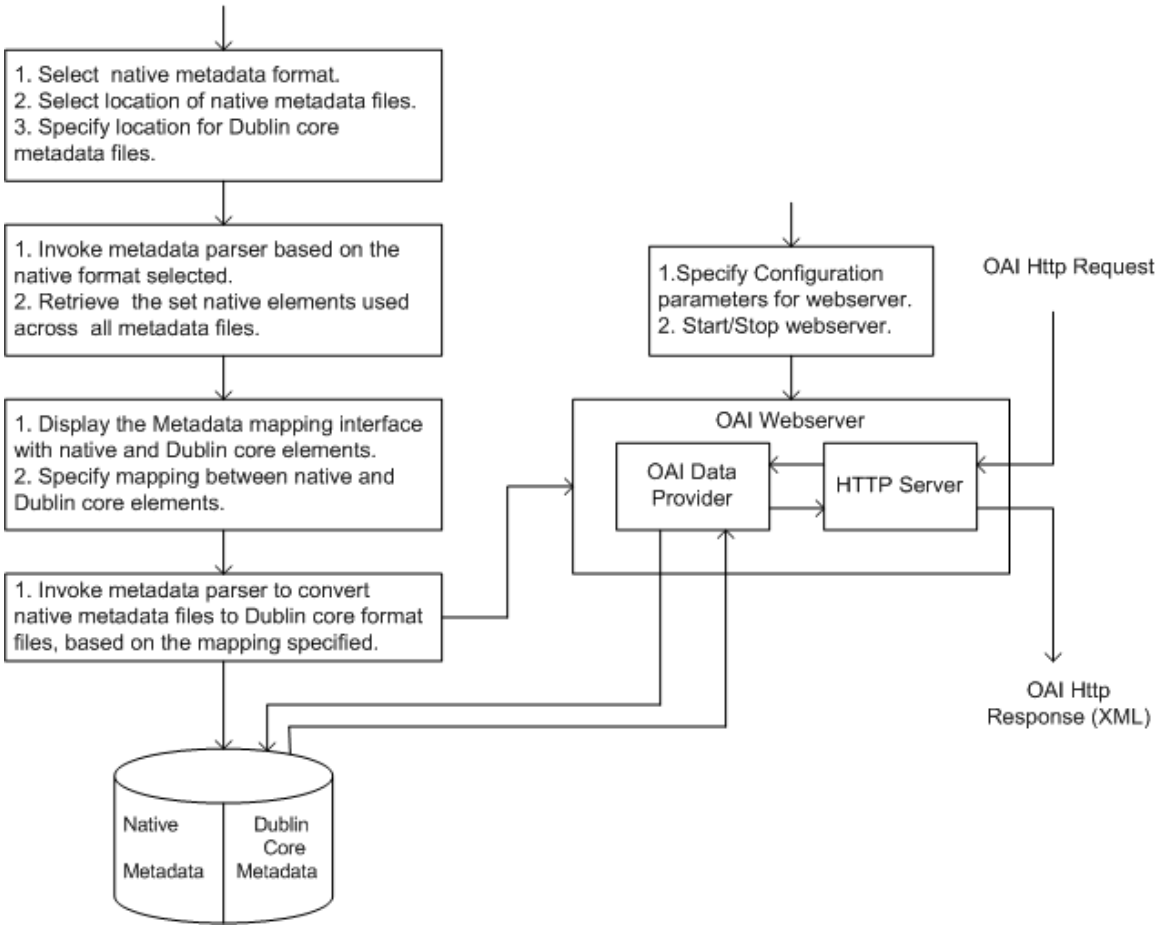

Figure 1. Metadata Conversion Process

Native to DC Mapping Definition Tool

The handling of the user's input (native format and location selection) to this module is provided using the GUI. Based on the selection, the metadata manager interacts with the appropriate Native to DC Metadata Converter and displays the metadata mapping interface with a set of native elements and DC elements. A mapping between native and DC elements can be then specified.

Native to DC Metadata Converter

This module consists of the Metadata Converter, along with the native metadata parsers. A parser is a program, which analyzes the native metadata, parses it and converts it to an intermediate format, which the Metadata Converter can understand. Depending on the native metadata format, the Metadata Converter uses one of the native metadata parsers to extract native metadata elements or to perform metadata translation. Once a mapping is defined, the Metadata Converter in this module uses the mapping and converts the native files to DC format files. This is the key module to make the tool extendable to support additional metadata formats.

Metadata Publishing Tool

This module supports publishing metadata in DC format. It has a user interface to add, modify, view and delete metadata. Once information corresponding to each DC element is specified using the user interface, this module validates the data, and the data is stored in the location specified for DC metadata files. Along with the data, full text documents are also uploaded to the DC metadata directory.

3.2 OAI Web Server

The Native to DC Metadata Converter module converts the native files to DC files. This satisfies one criterion for a repository to become OAI-PMH compliant. But, the repository should also have the capability to respond to OAI-PMH requests embedded in HTTP. The OAI webserver takes care of this issue.

The main objective of this module is to listen and respond to OAI requests. For this purpose the OAI web server is classified into two sub-modules

- Lightweight HTTP Server
- OAI –PMH Data Provider

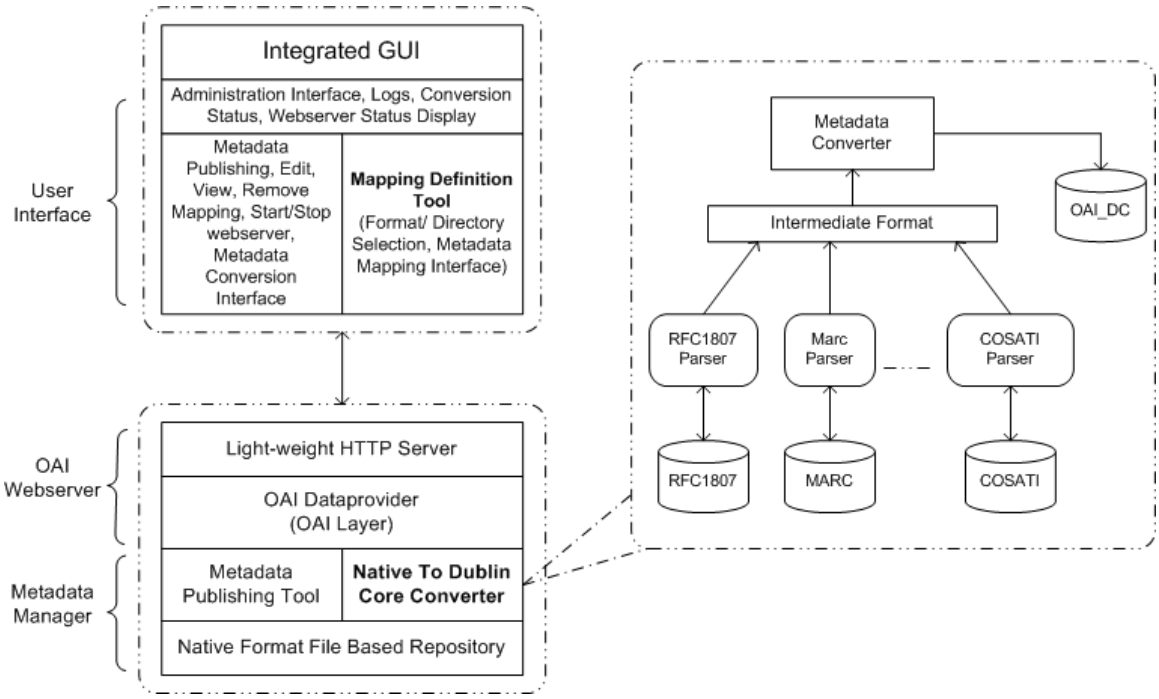

Figure 2. RVOT Architecture

Lightweight HTTP Server

This HTTP Server module performs the task of serving HTTP requests. It only accepts OAI-PMH requests embedded in HTTP requests and checks for protocol compliance. If the request is not protocol compliant it rejects the request, otherwise, it interacts with the OAI-PMH data provider module to generate the right response.

OAI-PMH Handler

This module is the core module of the OAI web server. It accepts the approved (protocol compliant) request from the lightweight HTTP server. From this HTTP request, the actual OAI-PMH request is extracted to get the exact OAI-PMH verb (action). Based on the action requested, it queries the base directory and creates a well-formed XML response. This XML response is then handed over to the HTTP server, which embeds it into an HTTP response and sends it out.

The request consists of a list of *keyword arguments*, which take the form of key=value pairs. Each request must have at least one key=value pair that specifies the OAI-PMH request. In the key=value pair that specifies the action, key corresponds to an OAI 'verb' and the 'value' is the value for 'verb' i.e. action to be performed. The number and nature of additional key=value pairs depends on the arguments for the individual request. The following are the list of verbs or actions supported by the RVOT's OAI web server.

- GetRecord
- Identify
- ListIdentifiers
- ListMetadataFormats
- ListRecords
- ListSets

If the request is invalid, then an exception message is sent to the client through the HTTP server, as appropriate a status code is returned.

Once the validation is complete and the verb or action is extracted, based on the action requested, the OAI-PMH data provider consults the base repository (directory) and constructs an OAI-PMH response. All responses to OAI-PMH requests are well-formed XML instance documents. The XML data for all responses to OAI-PMH requests are validated against the XML Schema specified by the OAI-PMH 2.0.

3.3 Integrated Graphical User Interface

The key for versatility of the RVOT is its ease of use through an integrated graphical user interface, which helps the user to perform various operations ranging from metadata publishing to OAI-PMH support. The graphical user interfaces provides for the following:

1. Metadata Mapping Definition Tool: This set of user interfaces help the user to make format selection, specify locations for metadata files, specify file extensions and to specify mapping between native and DC metadata elements. It also provides for conversion of native metadata based on a mapping specified earlier.
2. Metadata Publishing: This set of user interfaces allows the DL user to publish DC metadata. The DL user can add, edit, view and delete metadata. It also provides a facility to upload full text documents.
3. Administration: This set provides user interfaces for administrative tasks related to repository

management and the OAI web server. The administrative tasks include specifying administrator name, e-mail id, and OAI-PMH protocol version supported. The OAI web server tasks include starting and stopping the OAI web server, providing a base directory consisting of DC metadata files to serve OAI-PMH requests, and specification of a port number on which the OAI web server is to run.

4. Other User interfaces: This set of user interfaces display RVOT system logs, metadata conversion history, web server status and history and also provide online help to use the system effectively.

4 Extensibility

One of the key design features of RVOT is the ease of adding support for additional metadata formats. For this purpose, we introduced an intermediate format for representing any metadata format. As shown in the RVOT architecture diagram (Figure 2), the native format metadata parsers convert the native metadata into an intermediate format, which the tool can understand. Figure 3 shows the functionality of a parser. The tool uses the intermediate format data to perform further processing.

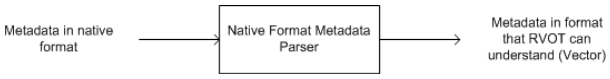

Figure 3. Function of Native Format Metadata Parser

The intermediate format is a *Vector* consisting of values of the format *'element=value'*, where element is the native metadata element and value is its corresponding value. So, no matter for which native format the parser is implemented, the job of the parser is to analyze the format, parse the metadata and construct the intermediate format vector and return it to the tool.

The tool uses this intermediate format for two purposes. One is to extract native metadata elements, to display on the mapping interface and the second is to convert the native metadata to DC metadata based on the mapping specified.

This approach makes the tool independent of the native format and thus makes RVOT extendable to other metadata format. The process of extending the tool involves implementing *Parser Interface* provided along with the package for the new metadata format.

The class diagram of the metadata manager as shown in Figure 4 illustrates the extensibility support (names in the diagram give indication of code reuse – Kepler project [10] – and metadata formats used, - RFC_1807 [7], MARC [3] subset). The RFC1807 parser handles metadata in RFC 1807 format. The MARC parser handles metadata used by NASA STI for their technical reports, and is a subset of MARC format. In later section, we describe how a developer can incorporate new metadata formats in the RVOT.

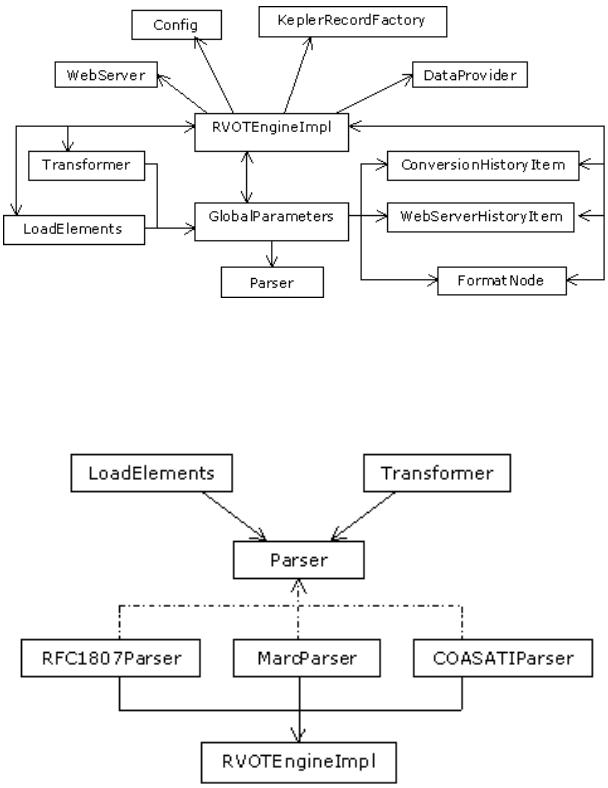

Figure 4. Class Diagram for Metadata Manager Suite

5 Sample Process of Metadata Conversion

In this section, we demonstrate the complete process as to how MARC format (a subset of MARC used by NASA STI) metadata is converted to DC metadata. Initially, using the Format/Directory selection interface, the MARC format is selected as the native metadata format and locations for MARC metadata and DC metadata are specified. The metadata converter module carries out a sequence of operations and the Metadata Mapping Interface is displayed. A mapping between MARC and DC elements is specified and once it is done, the metadata converter module carries out another set of operations and converts MARC metadata files to DC metadata files.

Figure 5 depicts the diagrammatic representation of converting MARC format to DC format. The MARC format parser converts metadata in MARC format to intermediate format. The intermediate format is used for extracting native elements and for metadata conversion.

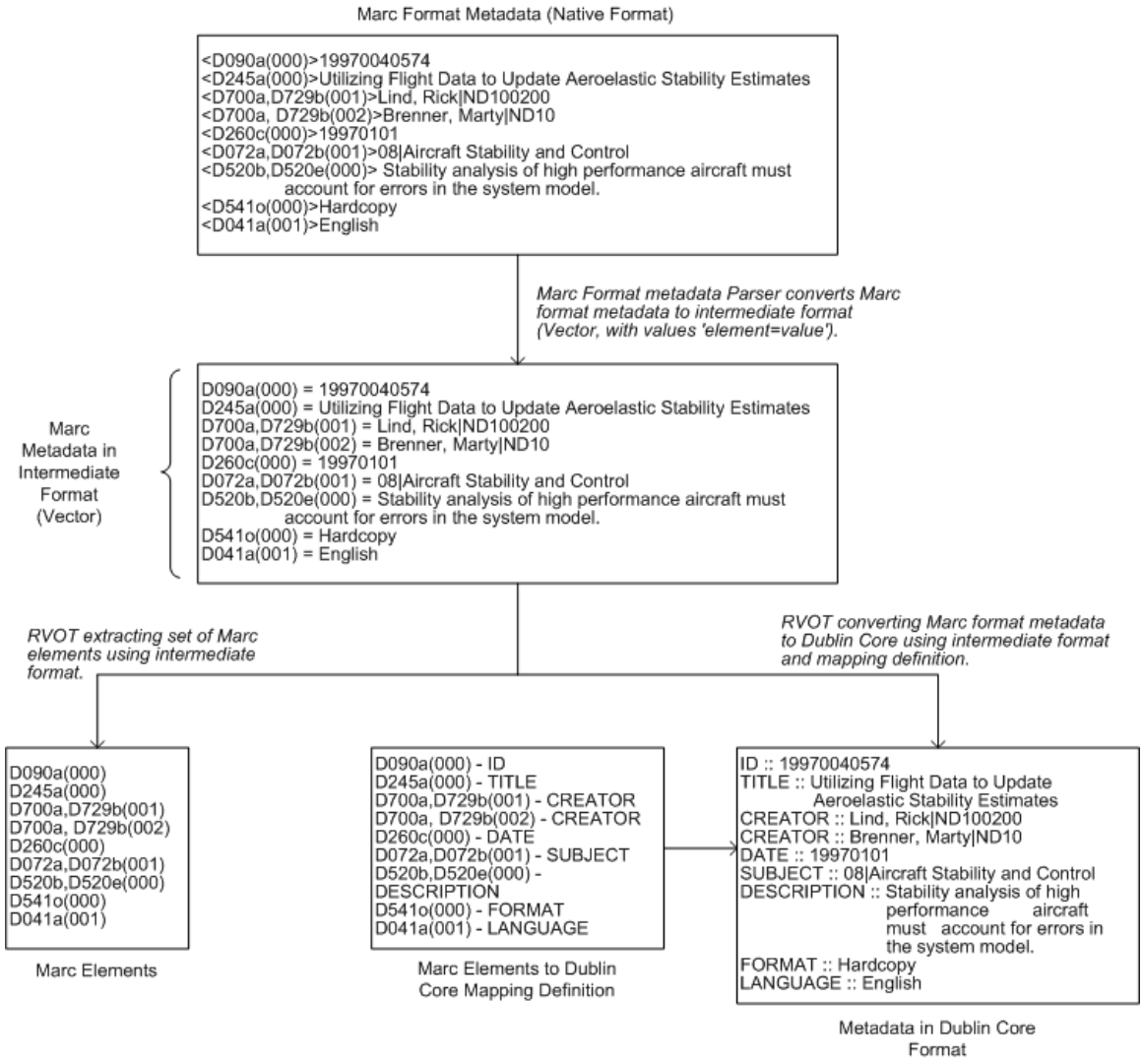

Figure 5. Translation between MARC and DC

6 Prototype

The system is written in Java and uses the native file system; it has a restricted web server and can be running on any machine with JVM (Java Virtual Machine) support. It does not require any other third-party packages such as RDBMS or web server, which makes the system easy to install and distribute. The downloadable package is available at http://rvot.sourceforge.net/.

The main interface (Figure 6) illustrates is the entry point to the major RVOT functionalities, including mapping availability, metadata conversion, and web server control. The user can follow the menus and run tasks in each sub-system. Figure 7 and Figure 8 show how to define the mapping table between different metadata formats. In Figure 8, a conversion is executed using the mapping defined earlier. Figure 9 shows the simple metadata publishing tool for DC.

7 Conclusion and Future Work

RVOT provides a *simple and effective* way to create an OAI-PMH compliant repository. We have completed the implementation and made the package available for use by the OAI community.

Table 2 summarizes the usage statistics for RVOT till date. These statistics are obtained from SourceForge.net website.

Table 2. Usage Statistics for RVOT

Lifespan	*Page Views*	*Downloads*
281 days	3266	123

Based on our initial release, we have received some encouraging and positive feedback and we have the following future plans for RVOT.

- At present RVOT supports native format files located on the local hard disk. We plan to enhance this by supporting URLs for the native metadata format files.

- Currently, the mapping definitions and DC metadata can be exported or imported by working with the file system i.e. accessing the appropriate directories and placing the mapping definition files or metadata files. This process of importing and exporting of mapping definitions and DC metadata files can be automated and can be done using the integrated graphical user interface.

- Currently, the process of exporting DC metadata is manual and it just allows DC metadata files to be copied (backup) to another location on the filesystem. The Export process can be automated not only to backup DC metadata but also to generate DC metadata in Static Repository format [13].

- Currently, to customize the tool to support other native metadata formats, the user has to write his/her own parser and place it in a particular directory and change some parameters in a particular file. The system can be enhanced such that the parsers can be generated automatically by taking some input from the user. Overall, this tool is semi-automated with respect to format, and this can be completely automated.

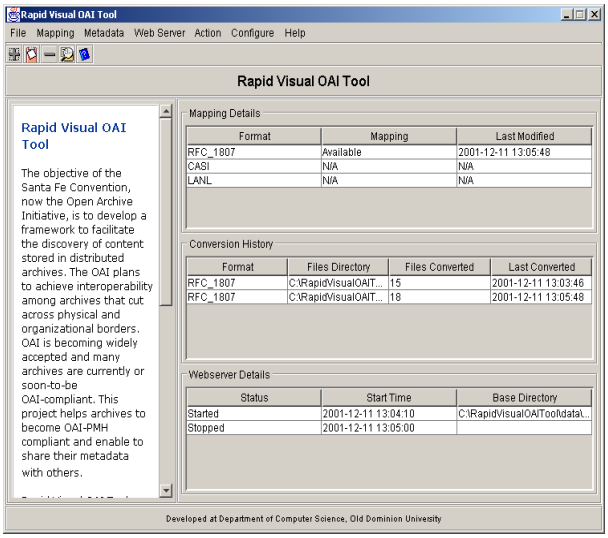

Figure 6. Main RVOT Interface

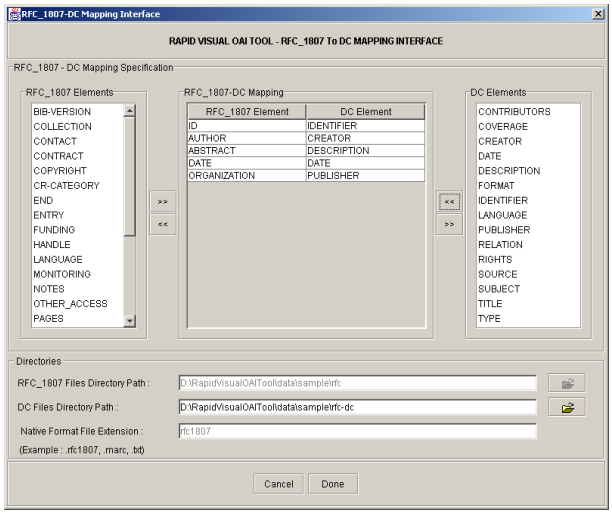

Figure 7. Mapping Native Metadata Format to DC

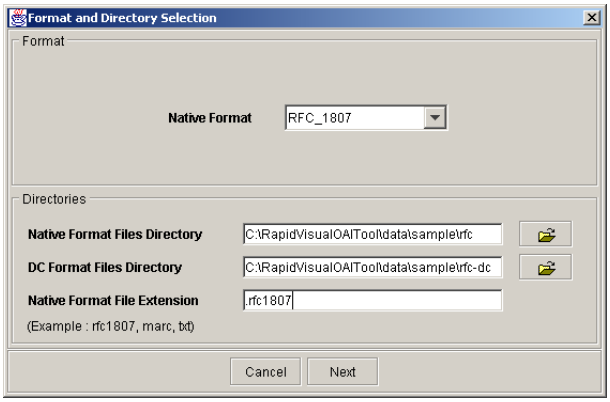

Figure 8. Location of Directories Interface

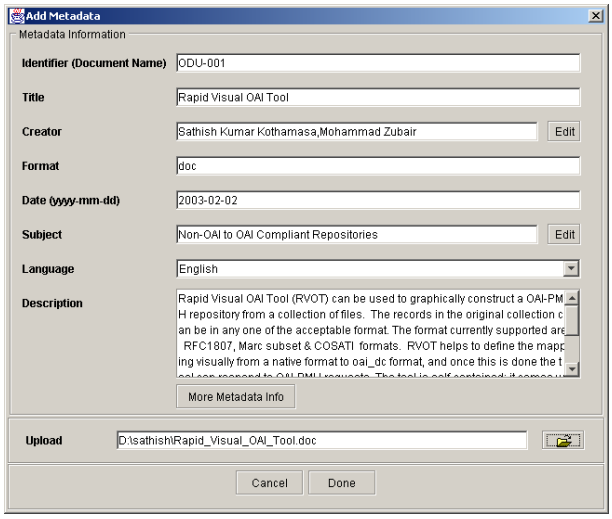

Figure 9. Publication Interface

References

[1] CERN Document Server Software (CDSware). http://cdsware.cern.ch.

[2] EPrints.org self-archiving FAQ. http://www.eprints.org/self-faq/.

[3] MARC Standards, Library of Congress. http://www.loc.gov/marc/.

[4] NCSA's COCOA (Components for Constructing Open Archives). ftp://emerge.ncsa.uiuc.edu/pub/cocoa/cocoa.jar.

[5] The Open Archives Initiatives. http://www.openarchives.org.

[6] The OAI Tools. http://www.openarchives.org/tools/index.html.

[7] D. Cohen, R. Lasher. "A format for bibliographic records". Technical Report Internet RFC 1807, IETF, 1995. http://www.faqs.org/ftp/rfc/rfc1807.txt.

[8] C. Lagoze, H. Van de Sompel, M. Nelson, S. Warner. "The Open Archives Initiative Protocol for Metadata Harvesting, version 2.0". http://www.openarchives.org/OAI/openarchivesprotocol.html.

[9] C. Lagoze, H. Van de Sompel. "The Open Archives Initiative: Building a low-barrier interoperability framework". In *Proceedings of the First ACM/IEEE Joint Conference on Digital Libraries*, Roanoke, VA, pages 54-62. http://www.cs.cornell.edu/lagoze/papers/oai-final.pdf.

[10] K. Maly, M. Zubair, X.Liu. "Kepler—an OAI data/service provider for the individual". D-Lib Magazine, 7(4), April 2001. http://www.dlib.org/dlib/april01/maly/04maly.html.

[11] M. Smith, M. Barton, M. Bass, M. Branschofsky, G. McClellan, D. Stuve, R. Tansley, and J. H. Walker. "DSpace - an open source dynamic digital repository". D-Lib Magazine, 9(1), January 2003. http://www.dlib.org/dlib/january03/smith/01smith.html.

[12] H. Suleman. "Enforcing interoperability with the Open Archives Initiative repository explorer". In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries*, pages 63-64, Roanoke VA, June 24-28 2001.

[13] H. Van de Sompel, C. Lagoze, M. Nelson, S. Warner, P. Hochstenbach, H. Jerez. "Specification for an OAI Static Repository and an OAI Static Repository Gateway". http://www.openarchives.org/OAI/2.0/guidelines-static-repository.htm.

[14] S. Weibel, J. Kunze, C. Lagoze, and M. Wolfe. "Dublin Core metadata for resource discovery". Technical Report Internet RFC-2413, IETF, 1998. http://www.ietf.org/rfc/rfc2413.txt.

[1] Throughout the paper a 'user' – unless specifically qualified - is the administrator within a small organization who makes its digital library OAI-PMH compliant.