Chapter 1
An Example-Based Generator of XSLT Programs

José Paulo Leal
University of Porto, Portugal

Ricardo Queirós
CRACS and ESEIG/IPP, Porto, Portugal

ABSTRACT

XSLT is a powerful and widely used language for transforming XML documents. However, its power and complexity can be overwhelming for novice or infrequent users, many of whom simply give up on using this language. On the other hand, many XSLT programs of practical use are simple enough to be automatically inferred from examples of source and target documents. An inferred XSLT program is seldom adequate for production usage but can be used as a skeleton of the final program, or at least as scaffolding in the process of coding it. It should be noted that the authors do not claim that XSLT programs, in general, can be inferred from examples. The aim of Vishnu—the XSLT generator engine described in this chapter—is to produce XSLT programs for processing documents similar to the given examples and with enough readability to be easily understood by a programmer not familiar with the language. The architecture of Vishnu is composed by a graphical editor and a programming engine. In this chapter, the authors focus on the editor as a GWT Web application where the programmer loads and edits document examples and pairs their content using graphical primitives. The programming engine receives the data collected by the editor and produces an XSLT program.

DOI: 10.4018/978-1-4666-2669-0.ch001

INTRODUCTION

Generating a XSLT program from a pair of source and target XML documents is straightforward. A transformation with a single template containing the target document solves this requirement, but is valid only for the actual example. Using the information from the source document, we can abstract this transformation. The simplest way is to assume that common strings in both documents correspond to values that must be copied between them. If we explicitly identify these correspondences, we can have more control over which strings are copied and to which positions. However, a transformation created in this fashion is still too specific to the examples and cannot process a similar source document with a slightly different structure. For instance, if the source document type accepts a repeated element and the example has repetitions of the element then the generated program would accept exactly repetitions of that element.

Although too specific, a simple XSLT program can be used as the starting point for generating a sequence of programs that are more general and are better structured, ending in a program with a quality similar to one coded by a human programmer. To refine an XSLT program we can use second order XSLT transformations, i.e. XSLT transformations having XSLT transformations both as source and target documents. In this approach, the role of an XSLT generation engine is to receive source and target examples, and an optional mapping between the strings of the two documents, generate an initial program and control the refinement process towards the final XSLT program.

The aim of this chapter is the presentation of Vishnu—an XSLT engine for generating readable XSLT programs from examples of source and target documents. Readability is an essential feature of the generated programs so that they can be easily understood by a programmer not familiar with the language. The architecture of Vishnu is composed by a graphical editor and a programming engine. The former acts as a client where the programmer loads and edits document examples and pair their content using graphical primitives. The latter receives the data collected by the editor and produces an XSLT program.

There are several use cases for an XSLT generation engine with these features. The Vishnu generator was designed to interact with a component that provides text-editing functions for the end-user or programmer. A client of Vishnu can be a plug-in of an Integrated Development Environment (IDE) such as Eclipse or NetBeans. In this case, the IDE provides several XML tools (highlighting, validation, XSLT execution) and the plug-in is responsible for binding the content of text buffers and editing positions with the engine and retrieving the generated XSLT program. Vishnu can also be used as the back-end of a Web environment for XSLT programming. In this case, the Web front-end is responsible for editing operations and invokes engine functions for setting the example documents and mappings, and retrieving the generated program. The generator can also be used as a command line tool as part of a pipeline for generating and consuming XSLT programs. In this last case, the generator processes example documents in the local file systems, making mostly use of default mappings.

This approach visual XSLT programming has obvious limitations. Only a subset of all possible XSLT transformations is programmable by pairing texts on a source and target documents. For instance, second order transformations or recursive templates are out of its scope. Use cases for Vishnu are formatting XML documents in XHTML and conversion among similar formats. For instance, creating an XHTML view of an RSS feed and converting metadata among several XML formats are among the possible uses of Vishnu. Moreover, we do not expect the automated features of Vishnu to produce the final version of an XSLT program. We view its final result as a skeleton of a transformation that can be further refined using other tools already available in Eclipse.