Automatic Construction of XML-Based Tools
Seen as Meta-Programming

BALTASAR TRANCÓN y WIDEMANN bt@cs.tu-berlin.de
MARKUS LEPPER lepper@cs.tu-berlin.de
JACOB WIELAND ugh@cs.tu-berlin.de
Technische Universität Berlin, Frankfurterstrasse 28/29, 10587 Berlin, Germany
(http://uebb.cs.tu-berlin.de/meta-tools)

Abstract. This article presents XML-based tools for parser generation and data binding generation. The underlying concept is that of transformation between formal languages, which is a form of meta-programming. We discuss the benefits of such a declarative approach with well-defined semantics: productivity, maintainability, verifiability, performance and safety.

Keywords: XML, SAX, DOM, TDOM, ANTLR, XANTLR, compiler construction, meta-programming, parser generation, data binding generation

1. Introduction

1.1. A possible land-map of XML applications

The simple fact that the bandwidth of communication channels and processors has increased rapidly during the last decade allows the revitalization of an old concept known from ancient UNIX days: the use of a simple thing like text as central medium for information interchange.

This is reflected by the common agreement upon the necessity of standardization, and the resulting acceptance and vivid participation in the XML standardization processes.

At first glance, these XML-related standards seem quite poor: The basic layer of the specifications (Bray et al., http://www.w3.org/TR/2000/REC-xml) just regulates the encoding of arbitrarily formed trees.

The next layer should give a notion of “type”. Here we find a diversity of different concepts: the ancient DTD, which is not expressive enough, the recently codified W3C-Schema, and more than half a dozen very interesting competitor schema languages,—each of them containing brilliant ideas and different nice features everyone would like to use cf. the survey given in Lee and Chu (2000), even newer is RELAX (Regular Language description for XML, http://www.xml.gr.jp/relax/), based on theoretical considerations in Murata et al. (2001).

But it is just because of the most simple notion of “text” the XML kernel imposes on its objects, just because of the absence of almost all typing restrictions and semantic implications, that XML based notations and tools potentially can (and probably will) infiltrate all areas of software engineering.
So the notions “XML-based encoding” and “XML-based architecture” will more and more get a significance like “ASCII-based”. Indeed the area of “XML-applications” does contain objects from totally divergent disciplines of software engineering,—each with very different underlying mathematical models, different traditions and ways of speaking, different grades of abstractions, etc. This total area could be described by a map with three landmarks in triangular position:

- One vertex of the triangle is made up by instances of XML used as a merely technically determined coding format for e.g.
  - tool configuration data, as in Trausmuth and Schneider (2001),
  - network protocol data units in client/server architectures, as in SOAP (Simple Object Access Protocol, W3C Note, http://www.w3.org/TR/SOAP), XLANG (Microsoft Corporation, 2001) etc.\(^1\)
  - database interfaces, representing both data and queries.\(^2\)
  - definition of meta-models, e.g. business items, e-commerce transaction objects,

- Most remote from that first vertex is the second one, defined by the needs of authoring, especially of “compound documents”. This is a very complex and inherently generic concept: (1) the integration of most heterogeneous materials (sound, business objects, graphics etc.) into a well-defined context must be supported, (2) for sake of re-usability and convenience mechanisms for parameterization of types as well as of documents seem highly desirable, and (3) such divergent applications as technical documentations (e.g. using the DOCBOOK DTD), scientific articles (in \LaTeX\ manner), cool web pages, table-oriented database views for web information services (HTML-like or ECMA-script backend) must be representable.

  This is the original field of SGML, and is covered in the authors’ toolkit by DDD (see figure 1, Lepper et al., 2001).

- Somehow related to both other areas, but requiring special treatment w.r.t. the correctness of transformations, is the usage of XML tree structures for the representation of “terms” of a given formal language. Some self-applications, like schema languages, use these kinds of semantics. Also the more elaborated species of the “coding formats” mentioned above are defined using simple grammars, and re-appear in this group.

  A promising approach on the level of system architecture for sake of inter-operability is the representation of “abstract syntax trees” (“ASTs”, which are used as output format of most parser codes) by XML structures.

1.2. Automatic parser generation by XANTLR and TDOM

While the authors have worked in all these three areas (cf. figure 1), the following presentation will concentrate on their work on this very last topic, the automated generation of an XML-based AST representation and its further processing by semantic transformations. The main focus is on two tools: XANTLR transforms annotated grammars to DTD and parser code (see Section 2), and TDOM constructs a typed document model for further processing (see Section 3).

While being usable independently, XANTLR and TDOM are designed to work in a pipeline. Indeed this combination has already been successfully used in one medium scale industrial