Chapter 42
Software Language Engineering with XMF and XModeler

Tony Clark
Middlesex University, UK

James Willans
HSBC, UK

ABSTRACT

XMF and XModeler are presented as technologies that have been specifically designed for Software Language Engineering. XMF provides a meta-circular, extensible platform for DSL definition based on syntax-classes that extend object-oriented classes with composable grammars. XModeler is a development environment built on top of XMF that provides an extensible client-based architecture for developing DSL tools.

INTRODUCTION

Software Engineering is different from other Engineering disciplines, such as Civil, Electrical and Chemical. Traditional Engineering disciplines are based on a collection of well-understood, fixed rules that are the same for each new system. There is a single *formal system* that describes the elements of each discipline and how they work together to build systems. A Software Engineer is free to design a formal system for each new application. For example, a financial application executes in terms of elements and rules that are different from a telecom system or a car engine controller. Each new formal system leads to a *domain specific language* (DSL) that can represent a family of related systems.

Conventional software development requires that systems are implemented in a *general purpose language* (GPL) such as C or Java. A *language engineering approach* requires that the DSL is embedded in a GPL. Fortunately, even a minimal programming language has a surprising property that allows programs to be expressed as data to be processed and executed by another program. This meta-ability that allows programs to be represented as data and *vice versa*, both differentiates Software Engineering from other Engineering disciplines and makes it possible for programming languages to process other programming languages.

The process of building languages is termed *Software Language Engineering* and requires a meta-technology that can build and process languages. A DSL can be *internal* if it is assimilated

DOI: 10.4018/978-1-4666-6042-7.ch042

as part of the host language and *external* if it is stand-alone. Languages that support *Language Oriented Programming* (LOP) allow DSLs to be defined and to be integrated with the host language execution engine (whether external or internal). Languages for LOP offer a range of features that can process syntax from *concrete* to *abstract* and can embed new language structures into the execution cycle of the host language either by *desugaring* to the host language structures or by producing data that is subsequently processed by an interpreter written in the host language.

XMF and XModeler are tools that have been designed for Software Language Engineering. XMF is a programming language for LOP and XModeler is an IDE written in XMF on Eclipse for building XMF applications and domain specific modelling tools. XMF is bootstrapped and XModeler is constructed by using XMF-defined languages to control a small number of tool primitives for graphics, tree-browsing, property editing, menus and text editors.

XMF is bootstrapped using a self-describing meta-model that supports an arbitrary number of meta-class instances. The XMF meta-model includes higher-order functions (closures) that are extensively used to process syntax structures and provides a basic language based on an extension of the Object Constraint Language (OCL) that conveniently supports a range of list processing operations. The basic language provided by XMF can be changed by replacing the default grammar. Finally, the meta-model for XMF allows classes to be associated with extensible grammars to form *syntax-classes*.

A syntax-class defines how to transform delimited text (concrete syntax) into abstract syntax that is subsequently processed by the XMF execution engine. Once defined, a syntax-class can be used in any XMF program. It can be embedded as an internal language construct in expressions or included via an external file. Unlike many systems, grammars can refer to each other and new language constructs can refer to the host language thereby allowing the new construct to be interleaved with the host.

XMF and XModeler were developed as commercial products and successfully used on a range of customer projects including those for BAES, Citi-Group, Artisan Software and BT (Georgalas et al., 2004, 2005). They are both open-source[1] and form the basis of language engineering examples described in the widely cited e-books (Clark et al., 2008). These tools were independently evaluated (Helsen Ryman & Spinelli, 2008) as providing the highest-level of support for systems abstraction compared to other tools for software engineering and is regularly included in comparative studies of language engineering tools.

XMF and XModeler provide key technical solutions when implementing languages for SLE and LOP. XMF has been the basis for a number of DSL developments (Clark & Tratt, 2010; Clark & Tratt, 2009; Clark et al., 2008; Clark et al., 2004; Petrascu & Chiorean, 2010). The key contribution that these technologies make to DSL development is to apply a uniform, reflective approach to language engineering. The meta-level architecture of XMF is completely open and extensible, and data at all levels (values, types, syntax, meta-types, operations) is represented using a simple single meta-circular representation. This leads to a uniquely uniform software language engineering technology that is described in this chapter.

SOFTWARE LANGUAGE ENGINEERING WITH XMF

Our proposition is that Software Engineering is predominantly a Language Engineering based discipline. The definition of any new system involves the comprehension of several domains, including the problem and solution domains, and the ability to control one or more technology platforms so that they support a collection of desired computations.