Science of Computer Programming 96 (2014) 396–416

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

A tutorial on metamodelling for grammar researchers

Richard F. Paige [*], Dimitrios S. Kolovos, Fiona A.C. Polack

Department of Computer Science, University of York, Deramore Lane, York, YO10 5GH, United Kingdom

HIGHLIGHTS

- We provide a tutorial introduction to metamodelling.
- We compare metamodelling with basic grammar technology.
- We examine three distinctive examples of metamodelling.

ARTICLE INFO

Article history:
Received 22 March 2013
Received in revised form 20 March 2014
Accepted 5 May 2014
Available online 28 May 2014

Keywords:
Metamodelling
Models
Grammars
Bridges
Unification

ABSTRACT

A *metamodel* has been defined as: a model of a model; a definition of a language; a description of abstract syntax; and a description of a domain. Because of these varied definitions, it is difficult to explain *why* metamodels are constructed, *what* can be done with them, and *how* they are built. This tutorial introduces the key concepts, terminology and philosophy behind metamodelling, focusing on its use for *language engineering*, and expressed in a way that is intended to be accessible to researchers who may be more familiar with the use of traditional context-free grammar techniques. We highlight the main differences between metamodelling and grammar-based approaches, describe how to map metamodelling concepts and techniques to grammar concepts and techniques, and highlight some of the strengths and weaknesses of metamodelling via a set of small, but realistic examples.

1. Introduction

Grammarware is the collection of grammars and grammar-aware theories and software [23]. The definition is very broad, and includes context-free grammars, graph grammars, XML schemas, class dictionaries and other techniques. In this paper, we restrict our focus to a subset of grammarware techniques – those that exploit context-free grammars (e.g., Backus–Naur Form, traditional parsing and semantic analysis techniques); these should be familiar to a significant number of readers *Modelware*, by contrast, is the collection of models, metamodels, model-aware theories and software systems. At the heart of modelware – influencing its theories, practices, tools and applications – is metamodelling. This paper provides an introduction to metamodelling for grammar researchers, focusing on traditional grammar technologies like context-free grammars and parsers. Our specific focus is on how metamodelling is used for *language engineering*, e.g., for implementing editors and analysis tools for domain-specific languages, general-purpose languages, etc. We assume that such researchers are comfortable and experienced with defining and implementing grammars (e.g., using Extended Backus–Naur Form (EBNF)), parser generator tools, and grammar-based manipulation of languages (e.g., for compilation, analysis, extraction and comparison); advanced grammar-based techniques (such as graph grammars and attribute grammars) are out of scope for comparison

[*] Corresponding author.
E-mail addresses: richard.paige@york.ac.uk (R.F. Paige), dimitris.kolovos@york.ac.uk (D.S. Kolovos), fiona.polack@york.ac.uk (F.A.C. Polack).

http://dx.doi.org/10.1016/j.scico.2014.05.007
0167-6423/© 2014 Elsevier B.V. All rights reserved.

and consideration in this paper. We also assume that these researchers have little or no experience with metamodelling, but are interested in the terminology relevant to metamodelling, how metamodelling is done, and what can be done with metamodels. Such researchers may be motivated to find ways to explain their research to modellers, and may seek to better understand modelling research.

The entry barrier to metamodelling can be high, not least because of the cumbersome terminology, and an absence of standard definitions. In this tutorial, we aim to lower the entry barrier to metamodelling, by building on essential knowledge of grammars.

We start the tutorial in Section 2 with some basic definitions, illustrated with very small examples of metamodels, constructed in different languages, that will hopefully be accessible and reasonably familiar to grammar researchers. In Section 3 we discuss the different motivations that exist for constructing metamodels, and suggest a number of activities and tasks that are possible once metamodels have been constructed and implemented. We also explain a typical metamodelling process, which will help to clarify some of the key differences between metamodelling and grammar-based approaches to language design. In Section 4, we compare metamodelling and grammars in some detail, in terms of some basic terminology, key conceptual differences, and the strengths and weaknesses of modelware and grammarware. We also briefly explain how to map key concepts from modelware to grammarware, focusing on the technical level (i.e., how implementation concepts from metamodelling can be encoded as grammar concepts). Finally, in Section 5, we present three examples that apply the metamodelling process, showing the construction of metamodels, and illustrating how they may be used to solve a selection of problems.

This paper is a substantially extended and thoroughly revised version of a tutorial that was first presented in [28]. Besides being a complete revision of the text from that paper, additional material includes a simple comparison of metamodelling and grammar concepts and techniques, a mapping from metamodel concepts to grammar concepts, and an additional detailed example shows the use of metamodelling techniques in the complex systems domain.

2. Definitions and examples

To understand what is a metamodel, and to define it precisely, it is convenient to first define *model*. We take a broad interpretation of the term, as captured in the following definition.

Definition. A *model* is a formal description of phenomena of interest, constructed for a specific purpose, and amenable to manipulation by automated tools.

Let us consider each part in turn. Descriptions (used in the sense of Jackson [22]) are fundamental in software and systems engineering; a formal description is made according to rules that have been specified, and that can be checked against. A model abstracts from the real world; as such, some phenomena are considered to be in-scope, and others are out-of-scope. Descriptions are also *externalised* (i.e., they are not mental models, or what are sometimes called *representations*) and can be exchanged and shared between stakeholders. Many different descriptions can be constructed of the same phenomena; it is important to understand the purpose to which the description will be put. For example, an operational model of sensor behaviour may be appropriate for simulation or exhaustive state exploration, but it would be inconvenient for proof of certain properties (because an operational model may lead to a very large state space in, e.g., a model checker). Finally, the manipulation of models by automated tools encompasses automatable model management tasks such as model transformation, comparison or validation. Models can thus be of phenomena related to, for example, systems or software engineering, or experimental science, or other problems.

Given a model, when is it valid? For example, given a finite state machine diagram, how can we determine if it is a valid diagram? Parts of the *model validation* problem are conceptually identical to the problem of determining whether a sentence is valid according to a grammar. For a finite state machine diagram we would want to ensure that only valid symbols (rounded rectangles, arrows, labels) are used, and that only states are connected by transitions (for example). We need an equivalent to a grammar, for models; the equivalent is, at least informally, a *metamodel* (though as we shall see, metamodels can express simple validity properties that go beyond those expressible using BNF).

Many definitions have been provided of the term metamodel. Among key literature in the area are Bézivin's papers on software modernisation [2] and *On the Unification Power of Models* [3]; and Atkinson and Kühne's *Model-Driven Development: a Metamodelling Foundation* [1]. The Object Management Group (OMG) has published numerous metamodel-related standards, including its MDA Foundation Model [12] which includes the OMG definitions of metamodelling. As yet, there is no expert consensus on a precise definition of metamodel, and differences in terminology across the definitions promote confusion.

Since our purpose is to lower the entry barrier to metamodelling, we adopt a simple definition, which is compatible with other researchers' definitions, but expressed in an unambiguous way.

Definition. A *metamodel* is a description of the abstract syntax of a language, capturing its concepts and relationships, using modelling infrastructure.

A language may be general-purpose (e.g., UML, SysML) or it may be domain-specific (e.g., for computer forensics [30]) – the latter we term *domain-specific languages* (DSLs). Both types of languages (for software or systems engineering) have an