A Model Engineering Approach to Tool Interoperability

Yu Sun¹, Zekai Demirezen¹, Frédéric Jouault², Robert Tairas¹, and Jeff Gray¹

¹ Department of Computer and Information Sciences, University of Alabama at Birmingham
(yusun, zekzek, tairasr, gray)@cis.uab.edu
² AtlanMod (INRIA & EMN), Nantes, France
frederic.jouault@inria.fr

Abstract. The integration of various tools is a common requirement throughout the software development process. It is often desirable to consult several tools that perform similar functionalities in the same domain to obtain different perspectives and results to assist design and maintenance decisions. In many cases, tool interoperability requires the generalization of tool-specific data, which necessitates homogenizing the data such that intellectual assets can be shared through a common framework (e.g., the integration of results from various clone detection tools). This tool demonstration summary presents a software language engineering solution technique that uses Model-Driven Engineering to address tool interoperability. A specific focus of the paper is a demonstration of model transformation applied to the task of homogenizing different data formats among similar tools. The challenges of tool integration are discussed in the paper, along with a detailed case study that highlights the benefits of applying a model transformation solution to tool interoperability.

Keywords: Model Engineering, Tool Interoperability, Model Transformation, Domain-Specific Languages, AMMA.

1 Introduction

The ability to model the key characteristics of new tools, and to integrate them with a set of previously defined tools, can be very useful. Often, however, researchers independently develop similar tools to perform the same functionality within a particular domain. Each isolated effort defines a different semantic model and uses uncommon storage representations. Unfortunately, this poses a problem when it comes to the important issue of integration – the result is an inability to provide a seamless exchange between tools. For example, our own past work observed numerous tools within the domain of avionics fault analysis that had very much in common, but those tools could not share models. A solution was an integrated model and associated tool adapters that allowed data exchange among the tools [7]. The area of Enterprise Application Integration (EAI) has provided several additional technologies to assist in the tool integration problem across many domains, such as healthcare [19]. This tool demonstration paper summarizes our investigation into using concepts from software language engineering, in particular model transformation, to address the tool integration problem.
In tool interoperability, different standards and formats make software interoperability a challenge [2]. There are several existing approaches that can be adopted to overcome the limitations of tool interoperability. For example, a general approach that is based on traditional parsing and interpreting activities can be implemented with general-purpose programming languages. As a second example, XML-based interoperability has emerged as a popular choice for a generic exchange format between software tools. However, although this works relatively well when all considered tools use some form of XML, this kind of solution is not as convenient in other contexts (e.g., when context-free parsing of the storage format is necessary).

A third approach to interoperability is based on model transformation and is the focus of this paper: the different formats are captured as abstract definitions of data structures (i.e., metamodels), and transformation rules map from one representation to another. The AtlanMod Model Management Architecture (AMMA) [8] is a model engineering framework that may be used to build bridges between tools or Domain-Specific Languages (DSLs). Each tool or DSL is captured and represented as a coordinated set of models. The work described in this tool demonstration summary uses three of the main capabilities of AMMA: metamodeling with the Kernel MetaMeta-Model (KM3) [4], model transformation with the AtlanMod Transformation Language (ATL) [6], and projections to (i.e., extraction) and from (i.e., injection) other technologies (e.g., grammars, XML). Projections are especially useful in the context of model-driven tool interoperability because each tool typically uses a specific file format. Notably, AMMA provides the Textual Concrete Syntax (TCS) [5] tool to deal with context-free syntax.

Although our specific tool demonstration is focused on using AMMA as a solution strategy, we believe that the general concept can be used with most modeling and language engineering tools. The next section presents a case study that demonstrates how software language engineering (specifically model engineering and model transformation) can be used to assist in the sharing of results and data across tools from the same domain. A concluding section summarizes the paper by presenting lessons learned and pointing toward future work.

2 Visual Representation for Clone Detection

Code clones are blocks of statements that are duplicated in multiple locations of one or more programs. Programs containing code clones are prone to manifest problems in the maintenance phase of the software development process [13]. A number of tools have been developed to automatically detect code clones in programs, such as CloneDR [1], CCFinder [12], Simian [10], and SimScan [11]. These tools take the source program as input and generate mostly textual reports about the code clones that were detected. Figure 1 shows excerpts of the reports generated by three popular code clone detection tools. The reports record the location and length of each code clone. Depending on the size of the application that is analyzed, the corresponding clone report files could contain tens of thousands of lines of text.