Available online at www.sciencedirect.com
ScienceDirect
Journal of Visual Languages and Computing
17 (2006) 508–527
Journal of Visual Languages & Computing
www.elsevier.com/locate/jvlc

AutoGen: Easing model management through two levels of abstraction[☆]

Guanglei Song[a,*], Jun Kong[b], Kang Zhang[a]

[a]*Department of Computer Science, University of Texas at Dallas, Richardson, TX 75083-0688, USA*
[b]*North Dakota State University, USA*

Abstract

Due to its extensive potential applications, model management has attracted many research interests and gained great progress. To provide easy-to-use interfaces, we have proposed a graph transformation-based model management approach that provides intuitive interfaces for manipulation of graphical data models. The approach consists of two levels of graphical operators: low-level customizable operators and high-level generic operators, both of which consist of a set of graph transformation rules. Users need to program or tune the low-level operators for desirable results. To further improve the ease-of-use of the graphical model management, automatic generation of low level of operators is highly desirable. The paper formalizes specifications of low- and high-level operators and proposes a generator to automatically transform high-level operators into low-level operators upon specific input data models. Based on graph transformation theoretical foundation, we design an algorithm for the generator to automatically produce low-level operators from input data models and mappings according to a high-level operator. The generator, called AutoGen, therefore eliminates many tedious specifications and thus eases the use of the graphical model management system.
© 2006 Elsevier Ltd. All rights reserved.

Keywords: Model management; Graph transformation; Graph grammar; Visual programming; Schema interoperation

☆The work is partially supported by the National Science Foundation under Grant no. IIS-0218738.

*Corresponding author.

E-mail addresses: gxs017800@utdallas.edu (G. Song), jun.kong@ndsu.edu (J. Kong), kzhang@utdallas.edu (K. Zhang).

1045-926X/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jvlc.2006.10.004

1. Introduction

With the advance of Internet applications, interoperation among different formats is becoming critically important. Many approaches have been intensely researched to provide systematic solutions for manipulating heterogeneous data sources, such as peer-to-peer data management [1]. Manipulating enormous heterogeneous schemas, however, has been relatively a forgotten research area. These heterogeneous schemas, such as XML Schemas [2], RELAX [3], SOX [4], ER models, SQL schemas and so on, are used to define data sources, called *meta-data* or *data models*. Traditional approaches to manipulating these data models are manually specified or designed case by case for specific domains, i.e. object-at-a-time. Information engineers have to put much effort to program specifically for the data model applications concerning data migration, data integration and translation. Such processes are time-consuming and error-prone, and eliminate the possibility of reuse.

To reduce the programming effort, Model Management [5] is introduced to reconcile the painful process of manipulating heterogeneous data models. According to the vision paper [5], model-related applications can be composed by a sequence of atomic operations on data models, such as Merge, Match and so on. These atomic operations are defined as generic operators such that they treat data models as high-level data structures and therefore can be re-used in various domains. A model management system provides a set of high-level programming interfaces for applications to implement the atomic operations to save programming effort. Model management is the first effort to organize and generalize these operators to a systematic architecture. Conceptually, these model management operators have been applied to solve many classic meta-data problems successfully [6] and the first textural prototype system has been developed [7].

Given the operators provided by a model management system, users need to write a program to combine a series of operators to fulfill a specific task. Each execution process of a specific operator is transparent to the user and not customizable. Many usage scenarios, such as the motivating example of Melnik et al. [8], however have demonstrated that user interventions are constantly required and customizability is highly desirable for model management operators. Existing operators, however, are defined by text and transparent to users, and their implementations are hard-coded in the system. Little work has been done to improve the customizability and user interfaces of model management operators.

To improve the expressiveness and customizability of model management operators, we recently proposed a graphical model management framework [9] based on a graph grammar formalism, i.e. the Reserved Graph Grammar [10,11]. We also presented graphical definitions and representations of data models and mappings. Many data models, including ER models and UML models, are represented by graphs and others, including XML Schemas and SQL schemas, can easily be translated into graphs [7]. With intuitive representations for designers to communicate with each other, graphs are natural representations for data models. Graph transformation, as the theoretical foundation of visual programming languages, is capable of formally defining how graphs should be built and how they evolve [9]. The framework defines operations on data models through a set of graph transformation rules. These transformation rules are declarative and customizable.

The framework provides two levels of graphical operators, i.e. *low level* for end-to users or adjust and *high-level* operators for domain-experts to program. The two tier architecture

Download English Version:

https://daneshyari.com/en/article/523627

Download Persian Version:

https://daneshyari.com/article/523627

Daneshyari.com