Creating Domain-Specific Modeling Languages Using a By-Demonstration Technique

Hyun Cho

University of Alabama
Department of Computer Science
Box 870290, Tuscaloosa, AL, U.S.A.

hcho7@ua.edu

Abstract

Domain-Specific Modeling Languages (DSMLs) have been widely used in several domains (e.g., finance, combat simulation, and image manipulation) because they have been shown to improve productivity and quality by reducing the gap between domain abstractions and computational expression within specific domains. DSMLs are often designed by iterating complex and mundane language creation tasks that requires domain knowledge and language development expertise. To tackle the challenges of DSML development, this poster abstract outlines a new approach for specifying and generating the abstract and concrete syntax of a DSML based on user demonstration. The goal of the research is to develop the underlying science and tool support to enable end-users to assist in designing a DSML for their domain, while minimizing the typical mundane tasks of DSML development involving many accidental complexities.

Categories and Subject Descriptors D.3.3 [**Programming Languages**]: Language Constructs and Features – abstract data types, Frameworks.

General Terms Design, Economics, Reliability, Languages.

Keywords Domain-Specific Modeling Language; By-Demonstration; Flexible Modeling Tools

1. Dscription of Purpose

Domain-Specific Modeling Languages (DSMLs) are languages that support encapsulation and abstraction of a particular domain. Due to tailored notations for the domain, they offer substantial benefits such as rich expressiveness and shortened learning curves for non computer scientists who need to express some computational task in a specific domain [7].

As an example, consider report designers for embedded devices, who do not have language development expertise, but would benefit from a domain-specific language to capture the diagrams that they currently draw informally. The formats of reports slightly vary according to the characteristics of embedded devices and customer requirements, but much of the reports are common. The job responsibility of these designers is to customize report formats for each embedded device and link each data field with predefined APIs. However, it may be challenging for them to implement quality report formatting because report formats may be hard-coded with a specific programming language. Therefore, the report designers may desire a DSML that can customize a report format through a WYSIWYG editor. In general, at least two experts (domain expert and language development expert) collaborate when creating a new DSML because learning language development techniques in a short time is difficult for domain experts who do not have language development expertise and hiring experts that have both domain and language development expertise may be challenging. Moreover, the complexity of the DSML development process makes DSML development tedious, error-prone and time-consuming. The purpose of the research described in this poster abstract is to provide a framework that assists domain experts, who have domain knowledge but do not have language development expertise, in building their own DSMLs.

2. Goals of the Research

Like other languages, a DSML consists of three main components; abstract syntax, concrete syntax, and semantics. The components are developed by iterating over complex language creation tasks until the language meets the needed requirements and/or reaches desired quality. To develop a DSML, the requirements of a problem domain are captured, and then the concrete syntax is defined by domain experts and language development experts. Based on the concrete syntax definition, language development experts define the abstract syntax (often represented as a metamodel). In addition, the semantics of a DSML is associated to the abstract syntax. Finally, domain experts and language development experts collaborate to verify the correctness of the DSML. As described, DSML development requires language development expertise as well as domain knowledge so that domain experts who do not have language development expertise may face several challenges if they try to develop their own DSMLs [2].

The main goal of the research described in this abstract is to build a framework that assists domain experts (who do not have language development expertise) in developing their own DSMLs. From the DSML development process, automated and systematic approaches for defining a DSML grammar (both concrete and abstract syntax) and its semantics are crucial toward resolving the challenges of DSML development. Thus, the research aims to simplify and automate DSML development by (1) capturing concrete syntax as end-users demonstrate their domain through modeling, (2) inferring abstract syntax from concrete syntax and model instances, and (3) associating semantics to abstract syntax.

Copyright is held by the author/owner(s).
SPLASH'11 Companion, October 22-27, 2011, Portland, Oregon, USA.
ACM 978-1-4503-0942-4/11/10.

3. Technical Approach

In this section, we describe the technical approach that is being pursued in this research. The overall process for developing a new DSML is represented by a framework for creating modeling languages that is shown in Figure 1.

DSML development begins with domain experts drawing their domain models with domain-specific notations in a modeling canvas of the framework. While domain experts demonstrate domain notions, the actions of these experts are recorded and analyzed to a candidate concrete syntax.

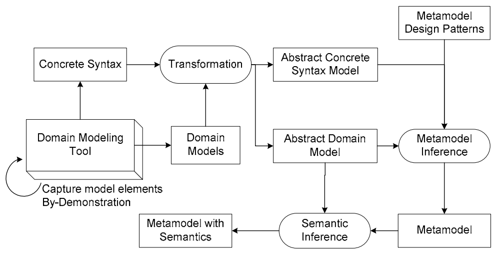

Figure 1. Framework for DSML

To infer the abstract syntax, the concrete syntax and domain models are transformed into a graph representation after the concrete syntax is identified. The inference process induces abstract syntax in the form of a metamodel by comparing abstract models of the concrete syntax and the domain models with patterns representing common metamodel idioms. Metamodel design patterns represent a set of training data that are passed on to an abstract syntax inference engine. The framework completes its process by inducing and associating relevant static semantics to the inferred metamodel.

3.1 Technical Challenges and Solutions

In this section, we identify major technical challenges that this research encountered. We describe the challenges and our solutions below.

- *Concrete syntax definition*: Concrete syntax represents the visual representation of a DSML through a set of domain notations and their relations. In our approach, candidate concrete syntax is automatically captured when domain experts demonstrate domain concepts using the notation of a specific domain. In order to capture the concrete syntax, we adopt a by-demonstration technique, which is similar to earlier work on Program By Example (PbE) or Query By Example (QBE) [4][8]. However, maintaining an optimized sequence of actions (i.e., pruning unnecessary actions) while preserving the demonstrator's intent is challenging.
- *Abstract syntax inference*: Inferring abstract syntax is a special case of inductive learning. This step aims to generate abstract syntax (i.e., metamodel for a DSML) by applying machine learning algorithms to concrete syntax and domain models, which are created in a previous by-demonstration step of the process. To induce abstract syntax, many approaches have been proposed, such as Bayesian learning [1], hill climbing [3], genetic programming [5], and graph [6]. In our research, we adopt a graph-based approach to leverage the benefits of graphs (e.g., analysis of structural and dynamic properties, design for verification, and model transformation).
- *Semantics inference*: Inferring semantics is emerging as the most challenging area of the research because specifying semantics in a formal way is difficult even for language development experts. Currently, the framework can infer some static constraints from model instances. To infer constraints, the semantic inference engine generates a model instances by traversing graph representation and obtaining feedback about positive and negative sample instances.

4. Current status and Future Work

A prototype has been developed with Microsoft Visio to understand the feasibility of the approach. The prototype was implemented with an understanding that Visio provides various predefined shapes that support the creation and definition of user-specific shapes. In this sense, a Visio template has a correspondence to the metamodel of a DSML. The prototype implementation focused on (1) the identification of concrete syntax, (2) creation of a Visio template, and (3) the identification of rules between shapes as a part of an initial static semantics.

Although we believe that the prototype demonstrates the feasibility of a "by-demonstration" approach for creating a DSML semi-automatically, the prototype is difficult to extend the functionality because Visio provides a limited set of APIs. Thus, we are working on extensions to the framework that can be applicable to other programming environments.

Acknowledgments

This work is supported by NSF CAREER award CCF-1052616.

References

[1] Chen, S. F. 1995. Bayesian grammar induction for language modeling. *In Proceedings of the 33rd annual meeting on Association for Computational Linguistics (ACL '95)*, Stroudsburg, PA, USA, 228-235.

[2] Cho, H., Sun, Y., Gray, J., & White, J. 2011. Key Challenges for Modeling Language Creation By Demonstration, *ICSE 2011 Workshop on Flexible Modeling Tools*, Honolulu HI, May 2011.

[3] Cook, C. M., Rosenfeld, A., & Aronson, A. R. 1976. Grammatical inference by Hill Climbing. *Information Science*. vol. 10, no. 2, 59-80.

[4] Cypher, A., Halbert, D. C., Kurlander, D., Lieberman, H., Maulsby, D., Myers, B. A., & Turransky, A. (Eds.). 1993. *Watch what I do: Programming by Demonstration*. MIT Press, Cambridge, MA, USA.

[5] Dupont, P. 1994. Regular Grammatical Inference from Positive and Negative Samples by Genetic Search: The GIG method. *Proceedings of the 2nd International Colloquium on Grammatical Inference and Applications*, ICGI'94, LNAI, vol. 862, 236-245.

[6] Jonyer, I., Holder, L. B., Cook, D. J. 2004. Mdl-based context-free graph grammar induction and applications. *International Journal on Artificial Intelligence Tools (IJAIT)*, vol. 13, no. 1, 65-79.

[7] Kelly, S. & Tolvanen. 2008. *Domain-Specific Modeling: Enabling Full Code Generation*. Wiley-IEEE Computer Society Press.

[8] Zloof, M. M. 1975. Query-by-example: the invocation and definition of tables and forms. *In Proceedings of the 1st International Conference on Very Large Data Bases (VLDB '75)*. ACM, New York, NY, USA.