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

Hyun Cho
hcho7@ua.edu

Advisor: Dr. Jeff Gray

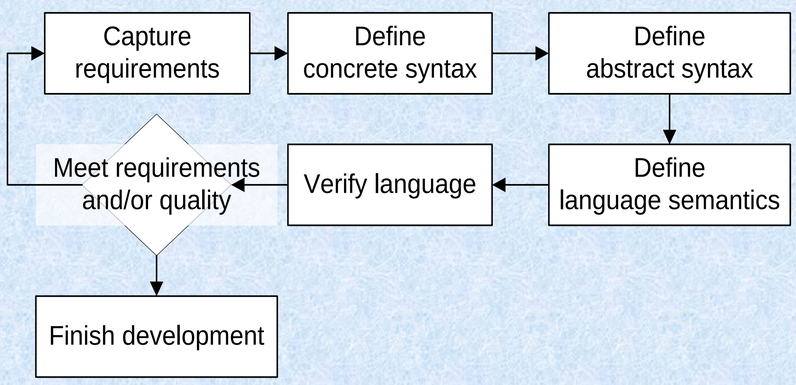
THE UNIVERSITY OF ALABAMA
COMPUTER SCIENCE

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

Although Domain-Specific Modeling Languages (DSMLs) have been applied successfully to various areas (e.g., finance, combat simulation, and image manipulation) and have shown improvements to productivity and quality, development of a new DSML is challenging for either computer scientists or end-users because it requires profound knowledge of the domain and deep experience in modeling language development. To address the challenges of DSML development, we propose a new approach for building DSMLs that represents a demonstration-based technique for specifying the details of a new modeling language.

Challenges in DSML Development

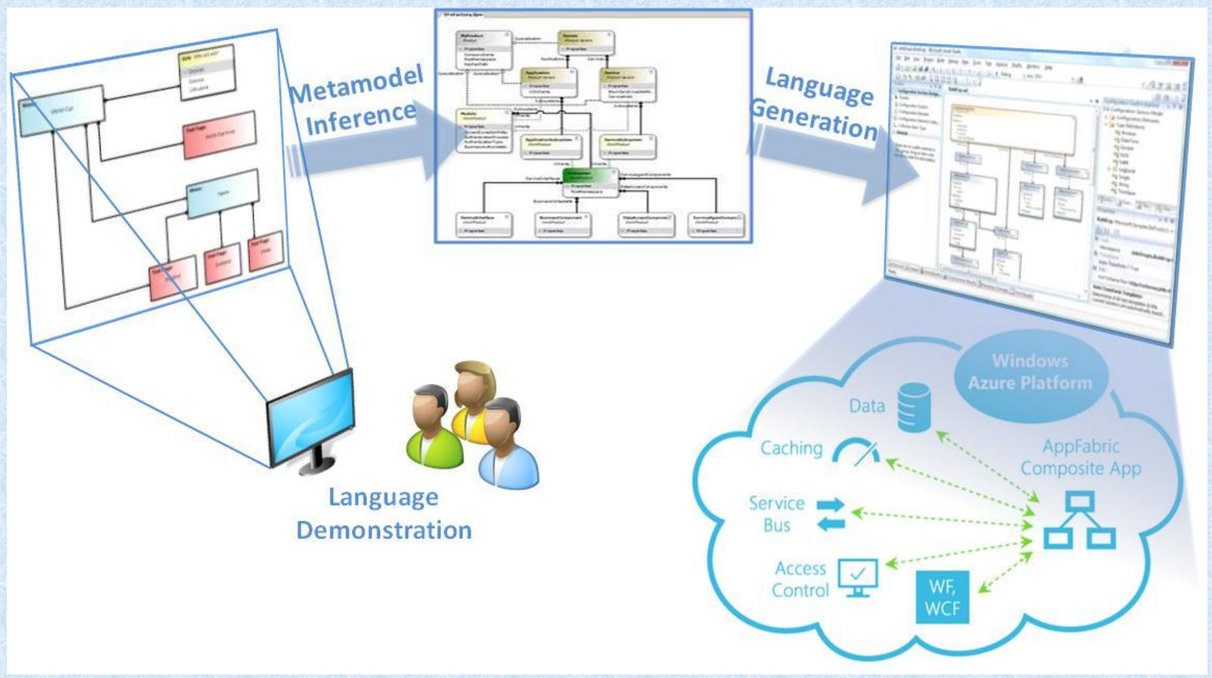
- **Domain-Specific Modeling Languages (DSMLs)**
  - ✓ Languages designed and implemented for specific domain requirements
  - ✓ Support encapsulation and abstraction of a particular domain
  - ✓ Provide notations tailored to the domain
- **Benefits of DSMLs**
  - ✓ Provide rich expressiveness
  - ✓ Minimize miscommunication
  - ✓ Shorten learning curve to use the language
  - ✓ Improve the quality
- **Barriers of DSML Development**
  - ✓ Domain knowledge and language development expertise are required when developing DSMLs, but few experts have such expertise
  - ✓ Lack of methods and guidelines to develop and manage quality DSMLs
  - ✓ Developed by iterating over complex language development tasks



Goal of the Research

- **Provide automated and systematic approach to domain experts who do not have language development expertise in developing their own DSMLs**
  - ✓ Capture concrete syntax of DSML automatically
  - ✓ Infer abstract syntax (or metamodel) from concrete syntax and demonstrated model examples
  - ✓ Associate semantics to abstract syntax

A Demonstration-Based Approach for Designing Domain-Specific Modeling Languages



Technical Challenges and Solutions

- **Support for free-form or sketch-level shapes**
  - ✓ Preference of domain experts to work in more unconstrained environments (i.e., whiteboard and paper-based sketch)
  - ✓ Advances in pen-based input devices
  - ➢ Shape recognition → Start with shape authoring tool and then embody free-form recognition functionality
- **Capturing concrete syntax**
  - ✓ Concrete syntax represents the visual representation of a DSML
  - ✓ Designed to avoid ambiguity and assist readability by domain experts
  - ➢ **Apply By-Demonstration technique**
    - Similar to Program-By-Example or Query-By-Example
    - Hook domain expert's actions when they demonstrate (or model) domain
  - ➢ Maintaining the optimized user action sequences without violating user's concerns → Design user action sequence management algorithm
- **Inferring abstract syntax**
  - ✓ Special case of inductive learning
  - ➢ Infer abstract syntax from a small number of model instance → introduce the notion of metamodel design pattern and devise new inference algorithm to utilize metamodel design pattern instead of a large set of training model instances
- **Inferring semantics**
  - ✓ Most challenging research area and under investigation

Current Status and Future Works

- **Developed prototype of the approach**
  - ✓ Development Environment: MS Visio SDK, Visual Basic Script, C++
  - ✓ Implemented functionality
    - Identification of concrete syntax
    - Automatic creation of MS Visio template
      → Correspond to abstract syntax inference
    - Identification and association of rules
      → Correspond to static semantics inference
- **Development plan for the general framework**
  - ✓ Phase 1: Preprocessor Development
    - Implement generic By-Demonstration functionality using Plug-in technology
    - Devise efficient algorithm to maintain the optimized user action sequences
  - ✓ Phase 2: Abstract Syntax Inference Engine Development
    - Using metamodel design pattern and Graph theory
    - Measure the success in terms of accuracy and performance
  - ✓ Phase 3: Semantics Inference Engine Development
    - Currently under development