Section 1: Week 3: Domain Specific Languages

Nate Bachmeier

TIM-8110: Programming Languages and Algorithms

May 19, 2019

North Central University

# Domain Specific Languages

A Domain Specific Language (DSL) is a mechanism for concisely describing the interaction with a well-defined context. Perhaps without knowing, even the most novice of computer programmers uses dozens of these languages.

A General-Purpose Languages (GPL) is a mechanism for describing a problem which spans multiple application domains. They tend to be more verbose than DSL due to needing to specify both the context and the interaction.

Consider building a simple webpage, which is written in HTML and CSS. These simple languages describe how a document should be structured and presented. When updates are ready for publishing the authors kick off a shell script, another DSL. Within that script regular expressions are used to transform place holders into final values in the configuration file. Both the RegEx and configuration file are also exampling of DSL.

The server-side code might be written in C# and use the ASP.net framework. To simplify the code behind development, programmers can write Razor templates to describe data binding scenarios in a mash of up XML and C#.

Both C# and XML are GPL languages as they are used across a wide range of software contexts such as client applications, automotive systems, and data science pipelines. The Razor templates are a DSL language as they are only used within ASP.net data binding scenarios. It does not matter that the Razor templates are using C# as their syntax and implementation. The key distinction here is the contextual use case.

# Categories of DSL

Domain Specific Languages can be categorized into distinct groups markup, modeling, and programming. These categories can be sub-divided into internal and external languages.

## Markup

Markup languages such as HTML, LaTex, and Markdown add metadata – such as formatting and font sizes, to a textual document.

## Modeling

Modeling languages describe an object hierarchy and their relationships. XML and JSON configuration files are common methods of persisting their representation. While it is perfectly acceptable to implement concrete DSLs within an abstract GPL, there are some draw backs. Most notably that the syntax is fixed and cannot be easily extended to add expressiveness.

This limitation can be mitigated by orchestrating grammar files through tooling such as JavaCC or ANTLR to construct micro languages. Perhaps an ancestry site uses this approach to expose the command ADD Jared AS BROTHER instead of <Add><Relationship Type=”Brother”><Name>Jared</Name></Relationship></Add>.

The ancestry site could also expose rich modeling syntax for traversing the lineage. Consider the command (me) > (parent) > (cousin[gender:male]) limit 10. This query finds up to 10 of my first cousins once removed. The readability of this statement within the context is sufficiently high that no additional details are needed.

## Domain Scripting

Domain Programming languages extend modeling languages to also add control flow such as branching and loops. TradeStation’s EasyLanguage allows business users to automate stock trading strategies. These users can specify IF current\_price < desired\_price THEN BUY 100 SHARES OF APPLE AT MARKTET PRICE.

The intent of that statement is instantly understandable to both the programmer and the domain expert. This clarity allows those experts to become more deeply integrated into the development cycle and ensure business rules are properly implemented.

## Internal vs External