Week 2: Domain-Specific Languages

Nate Bachmeier

TIM-7241:Constructive Research Design

May 30, 2021

Northcentral University

# Domain-Specific Languages

Domain-Specific Languages (DSL) enable business professions to describe their rules and policies succinctly. This capability reduces the barrier to automation, promotes cross-disciplinary collaboration, and modulizes complex systems. For instance, financial trading applications contain an execution engine and various investment strategies. These different components require unique professional expertise, making it challenging to have one person build both. Instead, the engineering team can standardize the problem’s shared vocabulary into external files that drive the trading engine. Afterward, domain experts can contribute to those files (scripts) and audit the accuracy of the trading algorithm.

## Recent Use-Cases

Countless scenarios can leverage DSL to automate specific business domains. For example, Cacciagrano & Culmone (2020) built IRON to simplify programming embedded systems through an Event-Condition-Action (ECA) model. The asynchronous nature of embedded systems is challenging to describe in a generic language (e.g., Java). Also, the operators are generally lightweight functions that act upon non-standardized messages. They could mitigate those issues using JavaScript. However, distributed JavaScript functions are difficult to troubleshoot and are not verifiable at build time.

Vernotte et al. (2021) propose the False Data Injection Attach (FDIA) language for identifying security defects within Air Traffic Control (ATC) systems. Building test cases for ATC is incredibly tedious, necessitating an asynchronous programming model with planes moving in multi-dimensional space (3-D + time + control flags). These requirements cause a typical test case to have over 235 parameters. Instead, FDIA supports short scripts that describe the test scenario. Next, it compiles into Automatic Dependent Surveillance Broadcasting (ADS-B) protocol, a standard within ATC validation tools.

## Language Design Challenges

Table 1: Summary of Reviewed Papers

|  |  |  |
| --- | --- | --- |
|  | IRON (2020) | Air Traffic (2021) |
| Business Problem | Integrating event-driven embedded system | Validating air traffic control protocols |
| Why DSL | Need abstraction layer for heterogeneous topologies | Test cases have 235 parameters on average (e.g., 3-D + time) |
| Challenges | No standard integrations |  |
| Artifacts | * LUA Interpreter * Verifiable call graph * ECA design pattern | * Compiles to AIS/ADS-B * Adapter for FDI-Test Framework |
| Measuring Effectiveness | * Abstract analysis (set theory) | * 11196 unique test cases |
| Language Extensions | * ECA is standard within cloud computing | * Naval control protocols |
| Measuring Extensions | * Usefulness wrt, e.g., Lambda | * See: Kontopoulos (2018) * TIM 7140 / W. 6 |