Swimming DSL

Conceptual Design, Domain Modeling, and Language Architecture

Carla Fernanda González Mina
cf.gonzalezm1@uniandes.edu.co

Abstract

Swimming DSL is a domain-specific language designed to describe, analyze, and generate swimming training sessions in a way that feels natural to athletes and coaches, while still being formally precise and machine-analyzable. This document focuses on the design decisions behind the language, showing how real training practice directly shapes syntax, semantics, and architecture.

1 Motivation: Why a Swimming DSL?

When swimmers receive a training plan, it usually comes as a text message, a PDF, or even a photo of a whiteboard. Everyone *understands* it, but the understanding lives only in human heads. As soon as we want to answer questions like *"How hard was this session?"* or *"How much freestyle did I swim this week?"*, the format completely breaks down.

What motivated this project is a very simple idea: **swimming workouts already behave like a language**. They have structure, repetition, keywords, and rules that are never written down, but always followed. Swimming DSL makes those rules explicit.

The goal is not to replace how coaches think, but to *respect it* and turn it into something that can be reasoned about, analyzed, and even generated automatically.

2 Design Philosophy or How I Thought About This Language

This DSL was designed with a very opinionated mindset:

- If something feels unnatural to a swimmer, it does not belong in the language.
- If something matters in real training, it deserves a semantic representation.
- Readability beats clever syntax every single time.

This is why Swimming DSL looks closer to a training notebook than to a traditional programming language.

3 Core Abstraction: The Training Session

The `session` construct is the heart of the language. Everything lives inside a session, because that is how swimmers think: one session at a time.

```
session morning {
swim 400 m freestyle easy pace 120
}
```

A session is not just a container. Semantically, it represents a complete unit of load, recovery, and intention.

3.1 More Expressive Session Examples

A single training idea can be expressed in many equivalent ways depending on the coach's intention.

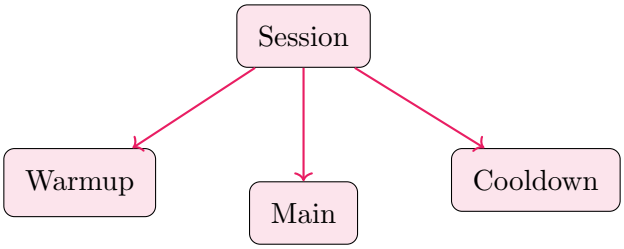
```
session aerobic_base {
  warmup {
    swim 300 m freestyle easy
    drill 4 x 50 m technique rest 15 s
  }

  main {
    6 x swim 200 m freestyle moderate pace 95 rest 20 s
  }

  cooldown {
    swim 200 m backstroke relaxed
  }
}
```

This example shows how the language naturally mirrors how swimmers already write workouts in their notebooks, while still exposing structure that can be analyzed.

3.2 Internal Structure of a Session



Sections are optional, but when present they add meaning. A hard set in warmup is interpreted very differently than a hard set in main.

4 From Text to Meaning: Syntax and Semantics

The concrete syntax was designed to feel almost conversational:

```
8 x swim 100 m freestyle hard pace 75 rest 15 s
```

Behind this simple line, the language captures repetitions, distance, style, intensity, pace, and rest as separate semantic entities.

4.1 Equivalent Expressions

Swimming DSL allows multiple syntactic forms that map to the same semantic meaning.

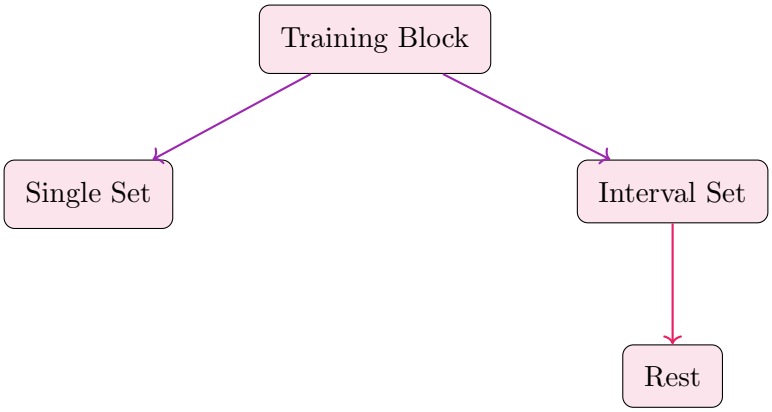
```
4 x swim 100 m freestyle hard rest 20 s
```

is semantically equivalent to:

```
interval 4 {
  swim 100 m freestyle hard
  rest 20 s
}
```

This design choice prioritizes readability for athletes, while keeping a single, clean semantic model underneath.

4.2 Block Semantics

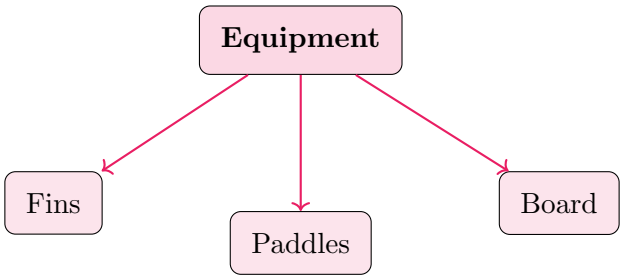


This separation is crucial for correct load and recovery calculations.

4.3 Equipment as Semantic Modifiers

In real training, equipment is never neutral. Swimming with paddles does not mean the same thing as swimming without them, even if distance and pace are identical. Swimming DSL models equipment as a semantic modifier rather than as a separate action.

```
swim 200 m freestyle moderate with paddles
kick 100 m easy with board
```



5 Domain Modeling: Styles, Intensity, Equipment

Swimming DSL treats domain concepts as first-class citizens.

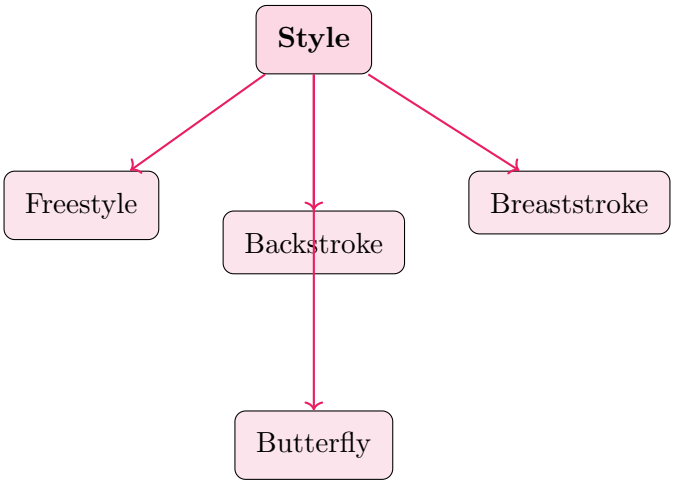
5.1 Typed Domain Concepts

Even though Swimming DSL is not explicitly typed in its surface syntax, internally the language treats domain elements as strongly typed entities.

```
Distance ::= meters(int)
Style    ::= Freestyle | Backstroke | Breaststroke | Butterfly
Intensity ::= Easy | Moderate | Hard
Rest     ::= seconds(int)
```

This separation prevents semantic errors such as assigning a rest value where a distance is expected.

5.2 Styles and Intensity



Intensity is modeled separately from pace, because effort and speed are related but not identical concepts in swimming.

6 Semantic Decisions

Some design choices are intentionally opinionated.

```
swim 100 m freestyle easy pace 90
```

In this case:

- `easy` affects perceived effort.
- `pace 90` affects expected execution speed.
- The language does not force a single source of truth.

This reflects real training practice, where pace and effort are related but not interchangeable.

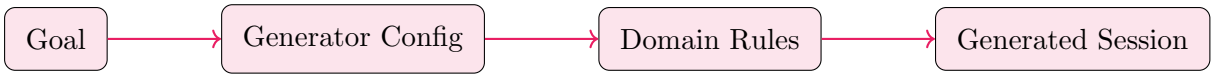
7 Automatic Generation: Teaching the Language to Think

One of the most powerful aspects of Swimming DSL is that it can *generate* sessions.

```
generate session {
goal: endurance
distance: 3000
duration: 60 minutes
}
```

This feature encodes coaching heuristics directly into semantics, turning the language into an active training assistant rather than a passive notation.

7.1 Generation Flow



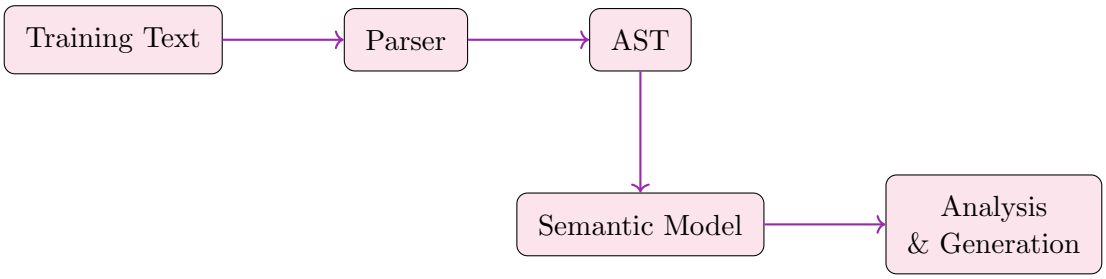
8 Full Semantic Pipeline

This pipeline clearly separates representation from meaning and interpretation.

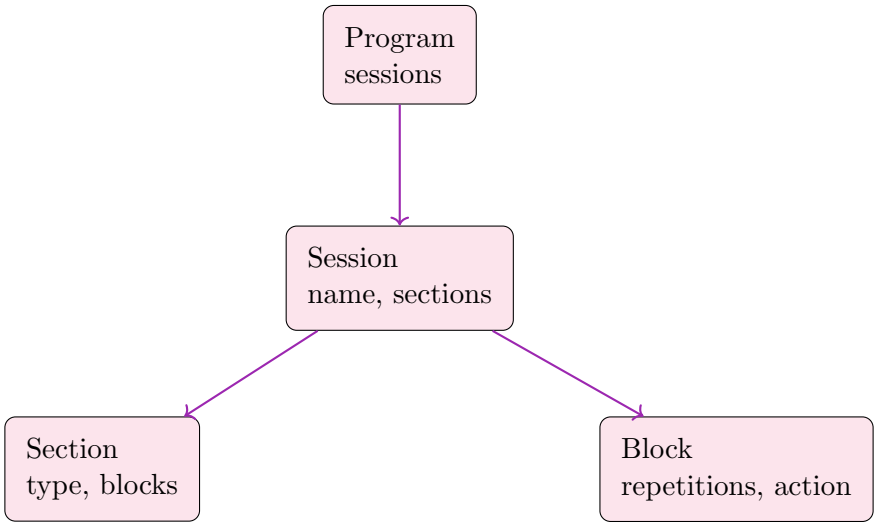
9 UML Architecture: Bringing Everything Together

After presenting syntax, semantics, and domain decisions separately, this section consolidates the entire language design into a single architectural view. The UML diagrams do not introduce new concepts; they simply formalize what has already been explained intuitively.

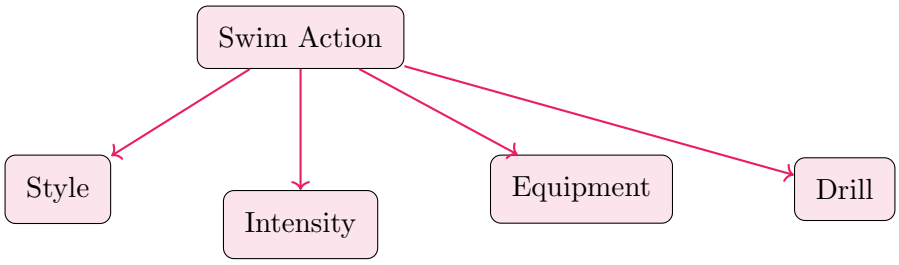
9.1 High-Level Language Architecture



9.2 Core AST Structure



9.3 Domain Model Relationships



10 Implementation Status

Swimming DSL is currently under active development. The repository contains the initial grammar, abstract syntax definitions, and early semantic analysis experiments.

GitHub Repository:
https://github.com/Carlixgonzam/swimmingdsl

11 Conclusion

Swimming DSL is not just a syntax experiment. It is a domain-driven language that respects how swimmers and coaches think, while enabling precise analysis and automation. The language shows how formal methods and real-world practice can meet without sacrificing usability or expressiveness. Of course made with love by Carlix.