# Introducción a los Lenguajes de Especificación de Dominio (DSL)

En todas las ramas de la ciencia y la ingeniería, se pueden distinguir dos enfoques: genéricos y específicos. Los enfoques genéricos proporcionan una solución general para los problemas de un área en particular, pero su solución puede no ser óptima. Un enfoque específico, proporciona una mejor solución para un conjunto pequeño de problemas.

En las ciencias de la computación, esta dicotomía existe con los lenguajes de programación general (GPLs) y los lenguajes de especificación de dominio (DSLs).

Un **lenguaje de especificación de dominio** (DSL) es un lenguaje de programación o un lenguaje de especificación ejecutable que ofrece, a partir de notaciones y abstracciones adecuadas, un poder de expresividad enfocado y por lo general restringido a un problema de dominio en particular [1].

**Lenguaje de especificación de dominio**, es un lenguaje de programación de computadoras, con expresividad limitada, enfocado en un dominio en particular [2].

De la definición anterior, se pueden destacar 4 elementos:

1. *Lenguaje de programación de computadoras*: Un DSL es usado por humanos para instruir a la computadora que realice cierta acción.
2. *Naturaleza del lenguaje*: Un DSL es un lenguaje de programación, y como tal, debe tener una sensación de fluidez, en donde la expresividad, debe originarse no solo de expresiones individuales, si no que también de la forma en que la mismas pueden ser compuestas como un conjunto.
3. *Expresividad limitada*: Un lenguaje de programación general proporciona muchas capacidades: soporte datos variables, control y abstracción de estructuras. Todo esto es útil, pero es más difícil de aprender y usar. Un DSL soporta, un conjunto mínimo de funcionalidades para soportar un dominio. No se puede construir un software entero con un DSL, en lugar de eso el DSL se concentra en un aspecto en particular de un sistema.
4. *Enfoque en el dominio*: Un lenguaje limitado, es solamente útil si tiene un enfoque claro en un dominio pequeño. El enfoque al dominio, es lo que convierte a un lenguaje limitado en en algo de real valor.

# Lenguajes de programación general vs lenguajes de especificación de

dominio

Todos los GPLs están conformes con las especificaciones de Turing, lo que quiere decir que pueden ser utilizados para implementar cualquier artefacto compatible con un máquina de Turing.

¿Por qué existen diversos GPLs? Todo depende de la estrategia de ejecución. El código *C* es compilato a un código nativo eficiente. Mientras que *Ruby* en ejecutado pr una máquina virtual (una combinación entre compiladores e interpretes).

Las características que ofrecen cada uno de los lenguajes, están optimizadas para las tareas que son relevantes a sus respectivos dominios. En *C* se puede manipular la memoria (aspecto importante para comunicarse con dispositivos de bajo nivel). Por otro lado *Ruby* se pueden utilizar "*closures*" para posponer la implementación de un comportamiento. La metaprogramación de *Ruby*, permite definir DSLs que son adecuados para el desarrollo de aplicaciones Web (El framework *Rails* por ejemplo).

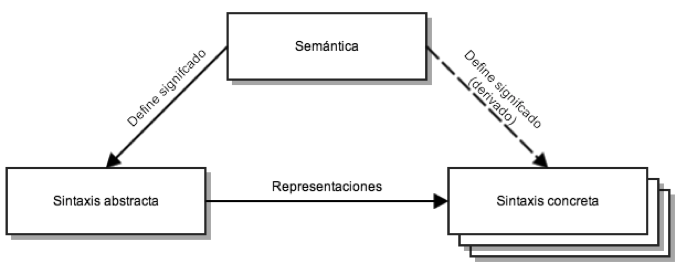
Incluso en el campo de los GPLs, existen diferentes lenguajes, cada uno proporcionado diferentes funcionalidades ajustadas a una tarea específica. Entre más específica se vuelve una tarea, es más adecuado el uso de lenguajes especializados. Un DSL, simplemente es un lenguaje que está optimizado para un clase de problemas en particular, llamados dominios.

# Ingredientes de un lenguaje de programación

Un DSL y un GPL, deben tener los siguientes ingredientes principales [3]:

1. *Sintaxis concreta*, define la notación que usarán los usuarios para definir los programas. Es la representación específica del lenguaje a modelar.
2. *Sintaxis abstracta*, es la estructura de datos que persiste la semántica de la información relevante expresada por un programa. Por lo general es un árbol o un gráfico. No debe contener detalles de notación.
3. *Semántica* describe el significado de los elementos definidos en el lenguaje y el significado de combinar dichos elementos.

La Figura 1, muestra la relación entre los tres ingredientes: la semántica define el significado de la sintaxis abstracta e indirectamente el de la sintaxis concreta; la sintaxis concreta representa la sintaxis abstracta.



"Figura 1. Ingredientes principales de un lenguaje"

Los DSLs se pueden clasificar en dos categorías:

1. *DSL externo*, es un lenguaje de programación que se construye, desde cero y tiene una infraestructura independiente para el análisis léxico, técnicas de parseo, interpretación y generación de código. [DSL in Action] (dichos conceptos son aborados en ???) .
2. *DSL interno*, es un lenguaje que utiliza la infraestructura de un lenguaje de programación existente; para construir la semántica de especificación de dominio encima de él. [DSL in Action]. (ejemplo: "Rails en Ruby")

## Diferencias entre lenguajes de programación general y lenguajes de especificación de dominio.

### Lenguajes de programación general

Cualquier computación puede ser expresada con una máquina de Turing. Los lenguajes de programación que cumplen con las especificaciones de Turing pueden resolver cualquier problema computacional. Dichos lenguajes son llamados: lenguajes de programación general (GPLs)

Entonces, ¿por qué existen diferentes lenguajes de programación?, ¿por qué no se puede programar todo en un solo lenguaje como Java, Python, Pharo o Ruby?, la respuesta está en el contexto de ejecución. Los lenguaje de programación, utilizan funcionalidades que están optimizadas para las tareas relevantes a su dominio (Es más conveniente desarrollar un sitio web en "Ruby on Rails", que en código C). Entre más específica es una tarea, más razón para especializar el lenguaje.

GPLs

DSLs

Domain

largo y complejo

pequeño y bien definido

Tamaño del lenguaje

largo

pequeño

Completitud de Turing

Siempre

No, por lo general

Abstracciones definidas por el usuario

Sofisticadas

Limitadas

Ejecución

Por medio de un GPL intermedio

Nativa

Tiempo de vida

Años/decádas

Meses/años (dependiendo del contexto)

Diseñado por

Guru or comité

Pocos ingenieros y expertos del dominio

Evolución

Lenta y estándard

Rápida

Depreciación/cambios incompatibles

Imposible, por lo general.

Factible

# Referencias

1. VAN DEURSEN, A., KLINT, P., AND VISSER, J. 2000. Domain-specific languages: An annotated bibliography. ACM SIGPLAN Notices 35, 6 (June), 26–36.
2. M. FOWLER. Domain-Specific Languages, Addison Wesley, 2010.
3. BOBADILLA M., CABOT J., WIMMER M., Model-Driven Software Engineering in Practice, Morgan & Claypool Publishers, 2012