Características de los lenguajes de programación
PyLP

361075 Carlos Eduardo Sánchez Torres
334908 Mauricio German De Los Santos Moreno

16 de febrero de 2021

Elegir la herramienta correcta gestiona la crisis del software e ilusionarnos por la tecnología de moda -esto es aplicable no solo para lenguajes-. Si entendemos los criterios, podemos predecir la adopción de esta o esa tecnología. Para evaluar un lenguaje de programación, primero entendamos en cual capa de abstracción nos encontramos, los requisitos y el mantra, filosofía o líneas guía son perseguidas por el autor (véase, por ejemplo, el Zen de Python [4] o de C++ [5]), para no caer en «si solo manejas un martillo, todo te parecerá un clavo». Si pensamos en interfaces web, pensamos en JavaScript, no en C. Si creamos un sistema SCADA, pensamos en lenguaje escalara, no en Python. Si lanzamos una aplicación para iOS será en Swift, no en Go. Esto viene a priori a cualquier lenguaje de programación: para decidir usarlo o los motivantes para crearlo.

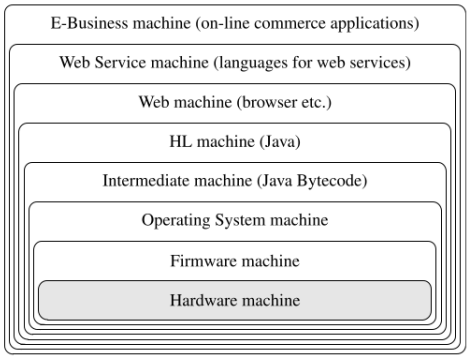

Figura 1: Capas de abstracción o de dominio [3].

Una vez dado unos criterio a priori, se definirá las métricas a considerar según [1] y [2].

1. Legibilidad

Antes de la crisis del software, los lenguajes se construyeron para ser lo mas rápido posibles, pensando principalmente en la computadora sobre el desarrollador. Después, de crear el concepto del ciclo del software como antítesis de la crisis del software, el foco ha sido la persona tras la pantalla sobre la computadora y el mantenimiento como parte central del ciclo.

La legibilidad es una métrica no binaria de los lenguajes de programación medida en términos del contexto de los problemas de dominio y su cercanía con el lenguaje natural. Así, decimos que un lenguaje es legible cuando naturalmente podemos leerlo, especialmente la persona de dominio y no tanto una técnica. ¿Usas más tiempo pensando como la máquina que pensando en resolver el problema de dominio?

Por ejemplo:

```
01001000 01100101 01101100 01101100 01101111 00100000 01110111 01101111 01110010 01101100
    01100100
```

Listing 1: «Hello World» en binario sobre ASCII. Ilegible.

```python
print("Hello World")
```

Listing 2: «Hello World» en Python. Legible.

A saber las siguientes características contribuyen a la legibilidad:

1.1. Simplicidad

Se refiere a la cantidad de conceptos o constructores disponibles en un lenguaje para construir el software requerido.

```c
typedef struct {
  int *array;
  size_t used;
  size_t size;
} Array;

void insertArray(Array *a, int element) {
  if (a->used == a->size) {
    a->size *= 2;
    a->array = realloc(a->array, a->size * sizeof(int));
  }
  a->array[a->used++] = element;
}
```

Listing 3: «Array dinámico» en C. No simple.

```python
a=[]
a.insert('A')
```

Listing 4: «Array dinámico» en Python. Simple.

1.2. Ortogonalidad

Los constructores se deben comportar de igual forma en todos los contextos. Los lenguajes deben ser precisos en su interpretación, es decir, se deben comportar siempre tal y como se espera que se comporten. Un lenguaje tiene generalidad eliminando casos especiales de los constructores.

1.3. Tipos de datos

Característica que tienen los lenguajes para que un programador no tenga que conocer muchos constructores para resolver el problema que está desarrollando. Es decir, un lenguaje tiene una capacidad de restricción si un programador puede centrarse en un subconjunto mínimo de conceptos del lenguaje para ser posible que construya la solución que necesita.

1.4. Sintaxis

Se refiere a lo parecido que se ve lo que se debe parecer, y lo distinto que se es lo que no debe ser igual: azúcar sintáctico.

2. Escribibilidad

2.1. Simplicidad y ortogonalidad

Es la capacidad de poder añadir nuevas características al lenguaje.

2.2. Expresividad

Es la cualidad de un lenguaje de programación que permite representar los procesos y estructuras complejos. Uso de las convenciones y notaciones estandarizadas, donde utilizan las notaciones y convenciones estandarizadas. Por ejemplo, el uso de operadores aritméticos básicos. Interoperabilidad, referido a la propiedad que tienen los programas de interactuar entre sí. Deben proveer mecanismos de intercomunicación, de forma que no haga falta comunicación previa para intercambiar información.

3. Fiabilidad

3.1. Revisión de tipos

Se refiere a la predicción de errores en el código fuente, evitándolos. Para hacer más seguros los lenguajes de programación se compromete en cierta medida la expresividad de estos.

3.2. Manejo de errores

Como gestiona los errores el lenguaje ante los fallos provocados por el contexto del sistema o por el usuario. Por ejemplo, en Go no existe el concepto de lanzamiento de excepciones.

4. Costo

Los apartados donde se destina dinero y tiempo necesarios para cumplir el ciclo de vida del software, tales como: curva de aprendizaje (ampliamente relacionado con la simplicidad), entrenamientos, escribir en dichos lenguajes (propósito de la aplicación, el mantra del lenguaje), ambientes de desarrollo (el lenguaje puede ser gratuito, por escribir en el no, por ejemplo Swift), costo de compilar (el tiempo necesario para llevarlo a producción puede llevar horas y eso se encuentra en un entorno de continua de integración puede ser bastante costoso), el costo de ejecutar dichos programas, el costo de implementar, costo de fiabilidad, costo de mantener los programas y la la posibilidad de construir programas independientes -portabilidad- de la máquina donde se ejecuten. Los lenguajes que se ejecutan sobre máquina virtuales son siempre portables: Java, Scala, PHP, ...

5. Soporte y comunidad

Lo relacionado con el mantenimiento del lenguaje y la búsqueda de respuestas para la implementación: licencia, foros, preguntas, librerías, frameworks, tecnologías alrededor, empresas y universidades, documentación, tiempo en el mercado, cantidad de proyectos y mantenimiento.

6. Julia

El mantra de Julia es ser tan interactivo como Mathematica, fácil de usar como Python y rápido como C: un lenguaje de programación para la ciencia [6] [7] [8].

6.1. Legibilidad

La cantidad de conceptos necesarios para implementar un algoritmo científico es mínimo: no es orientando a objetos.

```
"Hello World"
```

Listing 5: «Hello World» en Julia. Legible.

6.2. Escribibilidad

Codificar en Julia facilita bastante el trabajo del cientifico [6].

```
[1 2 3; 1 2 3]*I
```

Listing 6: Multiplicación de dos matrices. Escribible.

6.3. Fiabilidad

Es un lenguaje débilmente tipado. Es posible lanzar excepciones y gestionar los errores vía if, en vez de try and catch.

6.4. Costo

Un lenguaje gratuito, fácil de empezar a probar gracias a los notebooks en la nube. Sus tutoriales son principalmente en inglés. Para problemas científicos puede ser bastante útil, su ecosistema para desarrollo empresarial es limitado.

6.5. Soporte y comunidad

Licencia MIT: Julia es de uso libre y de código abierto. Con pocas librerías y un lenguaje relativamente nuevo (v1.5), en cualquier momento puede dejar de ser mantenido. Muchas universidades americanas están apostando al enseñar en la academia.

Referencias

[1] Tucker, A. B., & Noonan, R. (2007). Programming languages: Principles and paradigms. In Programming languages: Principles and paradigms (pp. 11-18). New York: McGraw-Hill Higher Education.

[2] Sebesta, R. W. (2016). Concepts of programming languages. In Concepts of programming languages (pp. 30-44). Upper Saddle River: Pearson.

[3] Gabbrielli, M., &; Martini, S. (2010). Programming languages: Principles and paradigms. In Programming languages: Principles and paradigms. London: Springer.

[4] 2021. PEP 20 – The Zen of Python. Python.org. Retrieved February 14, 2021 from https://www.python.org/dev/peps/pep-0020/

[5] The C++ programming language [Preface]. (2018). In 1195232291 893002481 B. Stroustrup (Author), The C++ programming language. Upper Saddle River i pozostałe: Pearson.

[6] Steven Johnson. MIT Applied Math MIT 337,. Retrieved February 15, 2021 from http://web.mit.edu/18.06/www/Fall17/1806/julia/Julia-intro.pdf

[7] Stephen Boyd. 2020. Introduction to Julia. Retrieved February 15, 2021 from https://web.stanford.edu/class/engr108/julia_slides/julia_intro_slides.pdf

[8] 2020. Julia Documentation · The Julia Language. Julialang.org. Retrieved February 15, 2021 from https://docs.julialang.org/en/v1/