Open Modular Interpreter

Análisis de mercado

Fco. Javier Bohórquez Ogalla

Índice

1. Introducción

En el presente documento se realiza una comparativa sobre los lenguajes de programación presentes en el mercado actual y sus principales características. El objetivo es determinar las carencias que estos presentan y que podrían ser cubiertas por el lenguaje de programación OMI. Para ello se tomará una muestra significativa y se estudiarán aquellas características consideradas de valor para el análisis.

Los lenguajes de programación tomados como muestra se corresponden con los más utilizados en el año 2014 según Gartner (consultor tecnológico estadounidense). Esta lista es utilizada como referencia en el mundo tecnológico por medios especializados.

- C/C++
- Java
- PHP
- Python
- Ruby
- JavaScript & Node.js (JS)
- OMI

2. Compilado o interpretado

Un lenguaje de programación es compilado si el código fuente desarrollado en el mismo debe ser traducido a código máquina para su ejecución. En contra partida un lenguaje es interpretado si el código fuente es procesado y ejecutado directamente por una pieza software denominado intérprete.

Muchos lenguajes son traducidos a un lenguaje intermedio denominado Bytecode que será ejecutado por un intérprete.

Algunos lenguajes interpretados pueden ser compilados a código máquina mediante alguna herramienta externas generalemente no estándar, estos casos no serán considerados en el análisis. De igual forma algunos lenguajes que normalmente son compilados pero que disponen de herramientas software capaz de interpretarlo.

	C/C++	Java	PHP	Python	Ruby	JS	OMI
Compilado	X						
ByteCode		X					
Iterpretado			X	X	X	X	X

3. Sistema de tipos

El sistema de tipos hace referencia a cómo el lenguaje clasifica los valores y expresiones en tipos. Este puede ser estático o dinámico en función si el tipo se establece en tiempo de compilación o ejecución.

Algunos lenguajes de tipado estático presenta mecanismos para escribir código que no será comprobado estáticamente, aunque estas implementaciones presentan ciertas limitaciones y pueden legar a ser ser inseguras en tiempo de ejecución produciendo resultados inesperados.

Por otro lado un lenguaje es fuertemente tipado si no permite usar un dato de un tipo concreto como si de otro se tratase, a menos que se realice una conversión explícita de tipos. Mientras que es débilmente tipado si no controla el tipo de las variables y demás datos que se utilizan. Normalmente un lenguaje de tipado dinámico es débilmente tipado.

	C/C++	Java	PHP	Python	Ruby	JS	OMI
Tipado estático	X	X					
Tipado dinámico			X	X	X	X	X
Tipado fuerte	X	X					
Tipado débil			X	X	X	X	X

4. Tipos de datos

Los tipos de datos que se pueden describir y manejar en los lenguajes de programación suponen un factor diferenciador de importancia entre estos.

Muchos lenguajes presentan tipos de datos que han sido modelados e integrados atendiendo al paradigma orientado a objetos. Sin embargo muchos tipos de datos pueden ser definidos de una forma más ágil o descriptiva usando un léxico y gramática propios.

Existen tipos de datos como las pilas, colas o árboles que son simplificaciones u otras implementaciones de otros tipos.

La solución que dan algunos lenguajes para determinados tipos de datos (como los

rangos) son en forma de otras construcciones del lenguajes como funciones u objetos.

Para profundizar en el tema se podría analizar la representación interna de los datos en cada lenguaje, para comparar atributos como la optimización de espacio o rangos permitidos.

	C/C++	Java	PHP	Python	Ruby	JS	OMI
Booleanos	X	X	X	X	X	X	X
Enteros	X	X	X	X	X	X	X
Flotantes	X	X	X	X	X	X	X
Cadenas de caracteres	X	X	X	X	X	X	X
Vectores	X	X	X	X	X	X	X
Vectores asociativos	X	X	X	X	X	(*)	X
Expresiones regulares		X	X	X	X	X	X
Nulo	X	X	X	X	X	X	X
Funciones	X	X	X	X	X	X	X
Objetos	X	X	X	X	X	X	X
Referencias	X		X				X
Símbolos o identidades					X		
Rangos			X	X	X	X	
Bloques de sentencias					X		
Tupla				X			
Tablas							X
Grafos							

(*) JavaScript realmente no presenta vectores asociativos, no obstantes estos pueden ser representados mediante objetos.

5. Paradigmas de programación

Los lenguajes de programación presentan cierto grado de enfoque en un determinado paradigma. Los hay así que son enfocados totalmente a uno único, tratando la mayoría de conceptos según ese criterio. Mientras que otros son multiparadigma, presentando así características, recursos y conceptos de varios de ellos.

El grado de enfoque en un determinado paradigma de un lenguaje de programación se puede determinar por la forma de tratar o modelar los distintos conceptos, y por la cantidad de recursos y características que presente a la hora de afrontar problemas según marco que el paradigma supone.

Los paradigmas no son independientes entre si, es muy común que un determinado paradigma se base en otros para su definición y tome parte de sus características. Se

consideran así los paradigmas más comunes o que supongan una base para otros más concretos.

Algunas de las características que serán analizadas pueden ser implementadas en determinados lenguajes mediante la concreción de otra más general o mediante el uso de otras más simples. Por otro lado existirán lenguajes que contemplen ciertos conceptos de una forma más directa, integrada en la gramática y más próxima a lo que persigue modelar.

Es común que algunos lenguajes implementen soluciones propias de un determinado paradigma pero en una forma y estilo propios de otro.

No formará parte del análisis aquellas características que son comunes en muchos lenguajes actuales: sentencias condicionales, sentencias iterativas, inclusión de ficheros, operadores con asignación, saltos a etiquetas ...

5.1. Imperativo

Describe la programación en términos de un flujo de instrucciones que operan sobre el estado del programa. Los lenguajes imperativos constan de sentencias que permiten controlar este flujo.

Existen paradigmas que toman el imperativo como base, por ejemplo la programación estructurada, por procedimientos o la modular.

	C/C++	Java	PHP	Python	Ruby	JS	OMI
Variables de ámbito global	X		X	X	X	X	X
Foreach		X	X	X	X	X	X
Iteración ágil(*)							X
Paso de parámetros por defecto	X		X	X	X		X
Keyword arguments				X	X		X
Número de parámetros arbitrarios	X	X	X	X	X	X	
Llamada sin respetar signatura						X	
Desempaquetado de parámetros				X			
Acceso a variables no locales				X			
Toda sentencia tiene un valor					X		

(*) La iteración ágil representa una sentencia de control de flujo iterativa cuya expresión es mínima en cuanto codificación se refiere. La sentencia unicamente se compone de una expresión que representa una secuencia de elementos y un bloque de sentencias. La expresión de secuencia será valorada según el tipo de dato que encierra. Así si la expresión es un vector se tomará como secuencia los elementos en el mismo, mientras que si es un

número se tomará la secuencia de números enteros desde 0 al mismo. El bloque de sentencias será ejecutado por cada elemento en la secuencia. En cada iteración el elemento correspondiente de la secuencia será asignado a un entidad propia del lenguaje denominada "elemento iterado" y la posición del mismo se asignará a otra entidad denominada "posición del elemento iterado". Estas entidades permiten el uso de un índice numérico para obtener el valor dentro de sentencias de este tipo anidadas.

5.2. Orientado a objeto

Describe la programación en términos de objetos que se relacionan entre si. Los objetos son estructuras que encapsulan un estado y una funcionalidad.

La programación orientada a objetos se basa en la programación imperativa.

Los lenguajes programación orientado a objetos normalmente utilizan clases para definir un conjunto de objetos que tendrán un comportamiento y estructura afín. Los objetos se consideran instancias o materializaciones de las clases.

	C/C++	Java	PHP	Python	Ruby	JS	OMI
Visibilidad	X	X	X		X		X
Definiciones estáticas	X	X	X	X	X		X
Polimorfismo	X	X	X	X	X	X	X
Duck typing			X	X	X	X	X
Herencia simple	X	X	X	X	X		X
Herencia múltiple	X			X			
Enlace estático dinámico			X				X
Interfaces		X	X				
Traits o Mixins			X	X	X		
Name mangling	X	X		X			
Métodos mágicos			X	X	X	X	X
Todo dato es un objeto				X	X	X	X

5.2.1. POO basada en prototipos

En los lenguajes basados en prototipos los objetos son creados directamente o mediante la copia de otros objetos. En estos lenguajes no se precisa del concepto de clase.

Los prototipos son objetos que serán clonados y de los cuales otros heredarán su comportamiento y propiedades.

	C/C++	Java	PHP	Python	Ruby	JS	OMI
Creación directa de objetos						X	X
Construcción de prototipos						X	
Herencia entre prototipos						X	

5.3. Dirigida por eventos

En la programación dirigidas por evento el flujo que sigue la ejecución de un programa viene determinado por los sucesos que ocurren en el sistema. Estos se pueden dar por acciones del usuario o por el propio programa.

Cabe decir que una programación dirigidas por evento no tiene porqué darse en un entorno de concurrencia. Un gestor de eventos denominado "event loop" puede operar sobre una cola de estos y ejecutarlos secuencialmente.

	C/C++	Java	PHP	Python	Ruby	JS	OMI
Detección de eventos						X	
Asociación de eventos						X	
Creación de eventos						X	
Lanzador de eventos						X	

5.4. Basada en autómatas

En la programación basada en autómatas el sistema se modela como una máquina de estados finita. Es un tipo de paradigma imperativo.

El concepto primario de este paradigma es el estado. Un estado captura un momento actual del sistema (t_0). Además a partir de un estado se deberá conocer todos los estados pasados ($t < t_0$), distinguiéndolos así de los futuros ($t > t_0$). A partir de un conjunto finito de estados y una serie de acciones de entradas se conforma un autómata.

Aunque con la mayoría de lenguajes imperativos se puede seguir este paradigma no es común que ofrezcan características enfocadas en el mismo.

	C/C++	Java	PHP	Python	Ruby	JS	OMI
Definición de estados							
Tabla de transiciones							
Construcción de autómatas							

5.5. Orientado a Aspectos

Describe la programación valiéndose del concepto de aspecto. Un aspecto es una funcionalidad que se presenta de forma transversal, ya sea en la totalidad del sistema o de una forma dispersa.

La programación orientada a aspectos persigue la correcta modularización del sistema, separando aquellas funcionalidades que son comunes a lo largo de toda la aplicación de aquellas que quedan encapsuladas en componentes.

Aunque es común ver la programación orientada a aspectos junto la orientada a objetos es posible aplicarla junto a otros paradigmas.

Aunque muchos de los lenguajes expuestos no soportan programación orientada a aspectos por si mismos, muchos de ellos constan de recursos externos que permiten la aplicación de este paradigma. Estos recursos amplían el lenguaje añadiendo reglas gramaticales y estructurales. En la mayoría de casos el código fuente debe ser preprocesado para obtener un código intermedio en el lenguaje base. Estos recursos no son considerados en el análisis ya que no se puede decir que el lenguaje contemple esta característica por si mismo.

	C/C++	Java	PHP	Python	Ruby	JS	OMI
Aspectos							
Puntos de corte							
Consejos							
Decoradores				X			X

5.6. Declarativo

Los lenguajes declarativos utilizan expresiones matemáticas para describir el problema y así obtener la solución.

En los lenguajes declarativos puros se cumple una transparencia referencial en todo el sistema por lo que se evitan efectos colaterales. Además no existen las asignaciones destructivas. Esto marca una diferencia con los lenguajes imperativos y es que las funciones declarativas no pueden depender o cambiar el estado del programa. Los lenguajes multiparadigma pueden ofrecer estructuras que garanticen estos principios.

Otros paradigmas son concreciones del paradigma declarativo.

	C/C++	Java	PHP	Python	Ruby	JS	OMI
Transparencia referencial							
Asignaciones no destructivas							

5.7. Funcional

Describe la programación en términos de funciones. Los lenguajes que atienden a estos paradigma generalmente son más cercanos al lenguaje matemático que al máquina. Es un paradigma declarativo.

Algunos lenguajes integran soluciones características de la programación funcional pero de una forma más próxima a la imperativa.

Cabe decir que algunas características funcionales pueden ser implementadas si el lenguaje dispone de otras características básicas, por ejemplo la currificación o la aplicación parcial pueden ser implementadas si se dispone de funciones de orden superior y de clausura. En estos casos no se considera que el lenguaje integre estas características de forma directa.

	C/C++	Java	PHP	Python	Ruby	JS	OMI
Función de orden superior	X	X	X	X	X	X	X
Clausura		X	X	X	X	X	X
Funciones Lambda	X	X	X	X	X	X	X
Pliegues			X	X	X	X	X
Mapeo de vectores			X	X	X	X	X
Filtro de vectores			X	X	X	X	X
Continuations			X	X	X	X	
Generadores			X	X	X	X	
Lista por comprensión				X			X
Currificación							
Aplicación parcial							X
Ajuste de patrones							
Guardas							
Composición de funciones							
Infijo para func. binarias							
Evaluación perezosa							
Mapeo y pliegue (x ==OR {1,2})							

5.8. Lógico

Describe la programación en términos de reglas lógicas que serán sometidas a un motor de inferencias para resolver los problemas planteados. Es un paradigma declarativo.

	C/C++	Java	PHP	Python	Ruby	JS	OMI
Hechos							
Reglas							
Motor de inferencias							

5.9. Programación por restricciones

La programación se describe en términos de restricciones que toda solución debe cumplir, luego el sistema se encarga de buscar la solución. La programación por restricciones se puede dar sobre distintos dominios de datos. Es un tipo de paradigma declarativo inicialmente derivado de la programación lógica.

	C/C++	Java	PHP	Python	Ruby	JS	OMI
Restricciones sobre booleanos							
Restricciones sobre enteros y racionales							
Restricciones sobre conjuntos finitos							

5.10. Programación concurrente

La programación concurrente ofrece soluciones para expresar paralelismo en la ejecución de tareas, permitiendo resolver problemas que se dan cuando varios procesos comparten o compiten por recursos.

La programación concurrente está muy presente en sistemas distribuidos y en interfaces de usuario.

La programación concurrente estudia y dispone de una serie de técnicas para la sincronización y comunicación entre procesos y/o hilos. Estas técnicas pueden ser implementadas o añadidas en forma de recursos externos en todos los lenguajes analizados, pero se estudia la integración de estas con la gramática del lenguaje.

	C/C++	Java	PHP	Python	Ruby	JS	OMI
Procesos	X	X	X	X	X	X	X
Hilos	X	X	X	X	X		
Mutex							
Semáforos							
Monitores							
Bloques sincronizados		X					

5.11. Programación distribuida

En la programación distribuida el sistema software se despliega en varias máquinas que pueden no estar próximas físicamente.

La programación distribuida ofrece soluciones a problemas de comunicación y sincronización entre sistemas que no se ejecutan bajo un mismo entorno.

La programación distribuida se produce en un entorno de concurrencia por lo que ambos paradigmas se encuentran muy relacionados. Ofrece una serie de técnicas para la comunicación entre procesos distribuidos mediante el uso de un canal de comunicación denominado socket.

	C/C++	Java	PHP	Python	Ruby	JS	OMI
Sockets	X	X	X	X	X	X	
Paso de mensajes							
Llamada a Procedimiento Remoto							
Modelos de objetos distribuidos							

6. Operadores

Los lenguajes de programación proporcionan una serie de operadores sobre los tipos de datos que define: lógicos, aritméticos, acceso...

En en el análisis se tomarán aquellos operadores que supongan un factor diferenciador.

	C/C++	Java	PHP	Python	Ruby	JS	OMI
Potencia			X	X	X		X
División parte entera				X			
Unión de array			X	X	X		
Intersección de array					X		
Diferencia de array					X		
Fusión de Nulos			X				X
Ev. cortocircuito booleanos	X	X	X				
Ev. cortocircuito último valor				X	X	X	X
Ternario reducido			X				X
Comparación combinada			X		X		

7. Funciones y clases de objetos

Los lenguajes generales ofrecen conjuntos de funciones y clases de objetos para afrontar problemas de muy distinta naturaleza. Estos pueden ser propios del lenguaje o ser añadidas mediante el uso de bibliotecas, módulos, extensiones...

Cada problema puede requerir soluciones propias. Los lenguajes de programación ofrecen gran variedad de recursos en forma de funciones o clases de objetos para dar solución a problemas comunes. No es objetivo de este análisis profundizar en este aspecto. No obstante cabe decir que existen determinadas categorías de recursos que pueden ser útiles en gran variedad de escenarios:

- Llamadas al sistema
- Fechas
- Ficheros
- Procesos
- i18n/l10n
- Bases de datos
- Matemáticas
- Sistema de ficheros
- Aleatoriedad
- Protocolos y formatos de internet

8. Definición de lenguajes específicos de dominio

Los lenguajes específicos de dominio son lenguajes enfocados a un problema o marco de problemas concretos. Existen lenguajes generales que permiten definir otros lenguajes destinados a un dominio específico.

Los DSL internos son lenguajes especificados a partir de la gramática y forma del lenguaje base sobre el que se escribe. La flexibilidad a la hora de escribir DSL internos depende en gran medida de la gramática y forma del lenguaje genérico. Existen lenguajes cuya gramática es muy flexible en este aspecto.

Los DSL externos son lenguajes que no están sujetos a la gramática y forma del lenguaje base. Debido a esto son más flexibles que los internos.

Aunque es posible escribir un DSL interno o externo en cada uno de los lenguajes referenciados, muchos de ellos no presentan características destinadas a tal fin y/o constan de ciertas limitaciones.

Es posible atribuirle a una gramática características como herencia, modularidad...

Existen lenguajes especializados en escribir gramáticas como bison o ANTLR.

El léxico de muchos lenguajes está conformado por cadena de caracteres que serán reconocidos, no obstante un lenguaje puede utilizar un léxico de otra naturaleza como sonidos, secuencia de eventos, grupos de píxeles, componentes de un diagrama...

Los DSL generados pueden ser creados e interpretados en tiempo de ejecución o añadidos mediante módulos compilados al interprete base.

	C/C++	Java	PHP	Python	Ruby	JS	OMI
DSL interno					X		
DSL externo							
Léxico flexible							

9. Depuración

Algunos lenguajes presentan características para la depuración de los programas que se escriben sobre ellos. Muchos pueden ser depurados mediante herramientas u otros recursos externos.

	C/C++	Java	PHP	Python	Ruby	JS	OMI
Asertos			X	X	X	X	
Depurador integrado			X	X	X	X	

10. Autodocumentación del proceso de interpretación

Un lenguaje puede tener la capacidad de autoexplicarse, por ejemplo puede dar información paso a paso sobre el proceso llevado a cabo para la interpretación, permitir seguir el flujo de ejecución o dibujar gráficas de rendimiento y espacio consumidos.

Para que el interprete/compilador presente autodocumentación debe ser capaz de dar información sobre todo el proceso. Además si integra mecanismo para generar DSL deberá documentar también los procesos aplicados a estos.

	C/C++	Java	PHP	Python	Ruby	JS	OMI
Análisis léxico							
Autómata léxico							
Análisis sintáctico							
Árbol sintáctico							X
Aplicación semántica							X
Árbol de análisis decorado							X
Estados de la tabla de símbolos							X
Tiempos de ejecución							
Espacio de memoria ocupado							X
Depurador integrado							X

11. Otras características funcionales

En este punto se presentan funciones que cumplen algunos lenguajes y que no han sido categorizadas, pero que aportan valor al análisis.

	C/C++	Java	PHP	Python	Ruby	JS	OMI
Constantes	X	X	X		X	X	
Expresiones con sigilo			X				
Extensiones			X	X	X		X
Reflexión		X	X	X	X	X	
Introspección de tipos	X		X	X	X		
Incrustable en otros documentos			X				
Acceso a estructuras de bajo nivel	X						
Docstring integrado				X			
Interprete interactivo			X	X	X	X	X
Excepciones	X	X	X	X	X	X	X
Espacio de nombres	X	X	X	X	X	X	
IPC integrado							
ORM integrado							X
Patrones de diseño integrados							X
Recolección de Basura		X	X	X	X	X	X
Principio de ocultación	X	X	X				
Niveles de aviso y errores			X				

12. Características no funcionales

12.1. Modularidad

Un interprete se puede presentar de forma modular según diferentes criterios:

12.1.1. Modularidad de recursos

Un lenguaje es modular en este sentido si tiene la capacidad de ampliar o configurar los recursos que brinda mediante módulos que encapsulan construcciones del lenguaje. Las construcciones (funciones y clases generalmente) encapsuladas en un módulo cumplen un propósito específico que puede ser útil en determinados problemas. Este tipo de módulos generalmente serán compilados de forma independiente y cargados dinámicamente, pero también pueden ser compilados junto el binario que representa el interprete.

La mayoría de lenguajes actuales son ampliable en recursos.

	C/C++	Java	PHP	Python	Ruby	JS	OMI
Modularización de recursos	X	X	X	X	X		X

12.1.2. Modularidad del lenguaje

El interprete puede tener una arquitectura interna modular, de forma que el lenguaje que representa puede ser extendido o personalizado tanto en léxico, como en gramática o en semántica. Estos módulos, que definen el lenguaje, pueden ser cargados en tiempo de ejecución o compilados junto al interprete.

Un lenguaje modular no es un lenguaje en si, sino un una base para construir otros lenguajes. A partir de un lenguaje de este tipo se pueden construir otros lenguajes, por ejemplo si se toma los módulos adecuados para construir el interprete se puede obtener un lenguaje puramente matemático.

Es necesario aclara que aunque el interprete tenga la capacidad de generar e interpretar lenguajes específicos de dominio, no quiere decir que brinde un lenguaje modular ya que las gramáticas generadas de esta forma se pueden ver como recursos del propio lenguaje. Un lenguaje capaz de generar DSL es modular en este sentido si el interprete que lo procesa puede compilarse sin esta característica.

	C/C++	Java	PHP	Python	Ruby	JS	OMI
Modularización del lenguaje							

12.2. Licencia

Todos los interpretes tomados en el análisis presentan licencias propias, pero es posible analizar las características de estas:

En este análisis se considerará únicamente la licencia de Node.js como JS dado que cada motor JavaScript dispone de su propia licencia.

Las licencias de C/C++ dependen del compilador tomado en el análisis, considerándose el compilador G++ de GNU.

Para Java se tomará la plataforma desarrollada por Sun Microsystems, aunque existen otras a las que se le aplican diferentes términos de licencia de uso.

	C/C++	Java	PHP	Python	Ruby	JS	OMI
Software libre	X		X	X	X	X	
Compatible con DFSG	X			X	X	X	
Aprobada por OSI	X		X	X	X	X	
Compatible GPL	X			X	X	X	
Copyleft	X				X		
Utilizable junto con otras licencias	X			X	X	X	

12.3. Rendimiento

El rendimiento supone una característica muy valorada en los lenguajes de programación. Debido a la gran cantidad de algoritmos que pueden ser codificados y la diversidad de estos, se hace difícil un análisis exhaustivo del rendimiento. No obstante existen una serie de benchmarks muy popularizados y estandarizados. Incluso se pueden ver proyectos dedicados a someter a los lenguajes de programación a un conjunto de estos benchmark (The Computer Language Benchmarks Game).

12.4. Productividad

Los lenguajes de programación deben perseguir la productividad de sus usuarios, para ello deben presentar:

- Léxico estándar, uniforme y sencillo.
- Gramática clara y directa.
- Estructuras estándar y uniforme.

Aunque todos los lenguajes analizados presentan estas características en mayor o menor grado, no es objetivo de este análisis entrar en discusión al respecto.

12.5. Portabilidad

Una característica común en muchos interpretes y compiladores es que se presentan como software multiplataforma, siendo posible su uso en la mayoría de sistemas operativos actuales.

La portabilidad es una característica muy valorada, pero que no supone un factor diferenciador dado que la mayoría lo cumplen.