Clase 8


framework
metrics
measurement
finance
key
individual
resources
important
directional
organization
team
goal
sales
process
achievement
potencial
metrics
definition

Clave

Las **Metodologías de Programación** son decisivas en el **éxito** o **fracaso** de un proyecto. En general ponen en práctica una serie de procesos comunes, que son buenas prácticas para lograr los objetivos de negocio, costos, funcionalidad, sencillez, etc. La elección de una metodología inadecuada o su mala aplicación pueden conducir a que el proyecto no llegue a su fin.

Tradicionales

Hasta hace poco se venían utilizando las denominadas **Metodologías Tradicionales**, donde los procesos son prácticamente secuenciales y están cargados de documentación, lo que los hace poco flexibles frente al cambio.
Hoy en día, con un escenario en el que los requisitos cambian habitualmente, es donde surge la necesidad de conocimiento sobre las **Metodologías Ágiles**, más ligeras y versátiles.

Elección

Es muy importante escoger una buena metodología para tener **éxito** en el proyecto, lo que implica muchas más cosas que sacar un producto a tiempo: supone que el cliente esté satisfecho y que el equipo trabaje cómodo. Actualmente, las **Metodologías Ágiles** están en auge dentro del desarrollo software.

1. Definiciones

Método

Procedimiento utilizado para llegar a un FIN.
Viene del griego μέθοδος, “camino que conduce a un lugar”.

1. Definiciones

Metodología

a. Ciencia del **método**.
b. Conjunto de **procedimientos** racionales utilizados para alcanzar los objetivos de una investigación científica, una exposición doctrinal o tareas que requieran habilidades, conocimientos o cuidados específicos.
c. Estudio o elección de un **método** adecuado para un determinado objetivo.

1. Definiciones

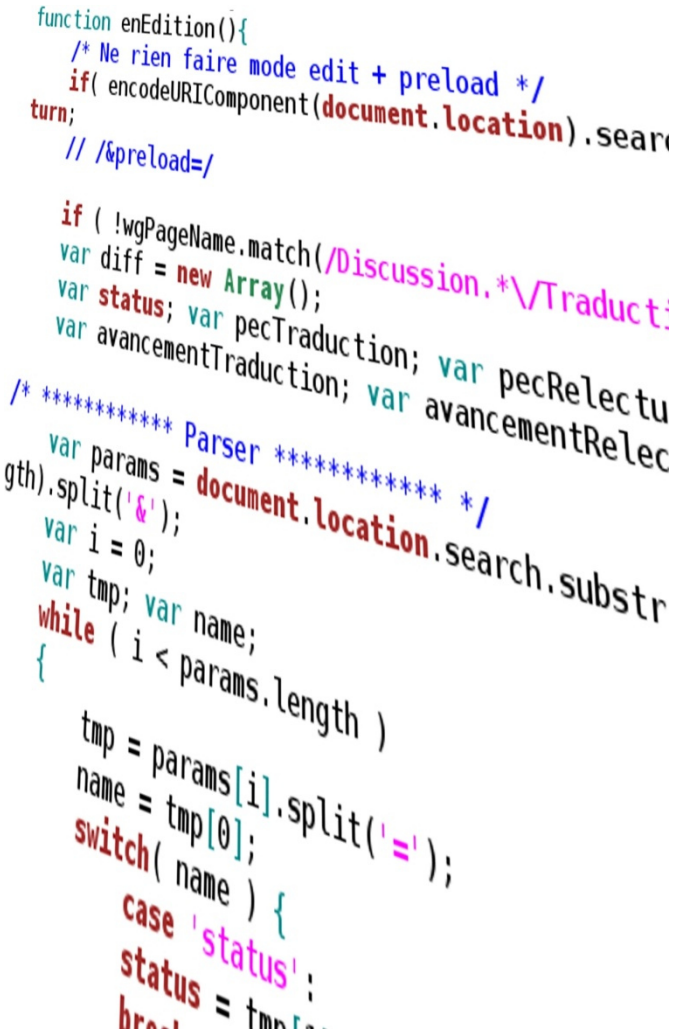
Software

Colección de programas, procedimientos, documentación y datos asociados que determinan la operación de un sistema de computación.

1. Definiciones

Programa

Conjunto de instrucciones que al ser ejecutadas permiten que una computadora realice sus diversas funciones.



```
function enEdition(){
   /* Ne rien faire mode edit + preload */
   if( encodeURIComponent(document.location).sear
turn;
   // /&preload=/

   if ( !wgPageName.match(/Discussion.*\/Traduct
   var diff = new Array();
   var status; var pecTraduction; var pecRelectu
   var avancementTraduction; var avancementRelec
/* ********** Parser ********** */
   var params = document.location.search.substr
gth).split('&');
   var i = 0;
   var tmp; var name;
   while ( i < params.length )
   {
      tmp = params[i].split('=');
      name = tmp[0];
      switch( name ) {
         case 'status':
         status = tmp
```

1. Definiciones

Programación

Proceso de diseñar, codificar, depurar y mantener el **código fuente** de programas de computación.

1. Definiciones

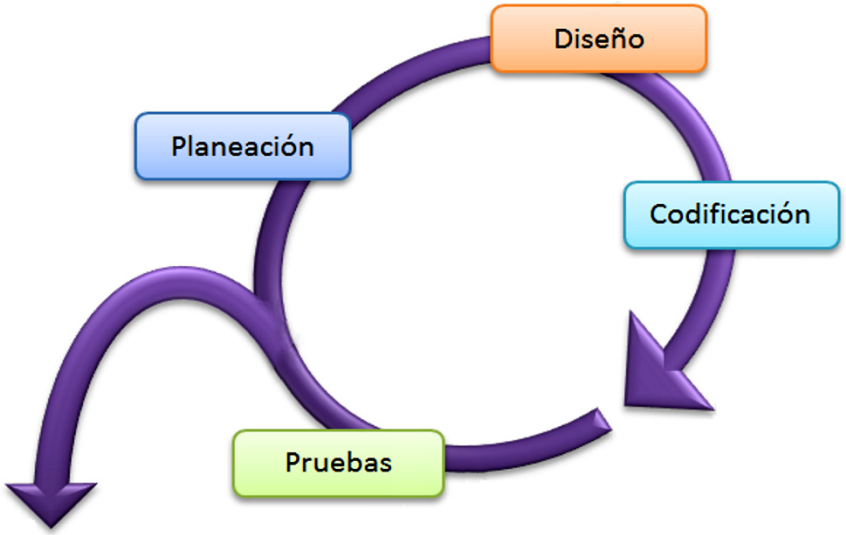
Framework

Conjunto estandarizado de conceptos, prácticas y criterios para enfrentar y resolver problemas de características similares.
Palabra del idioma inglés que significa “marco de trabajo”
Consiste en modelos, reglas, herramientas para asistir al proceso de desarrollo

1. Definiciones

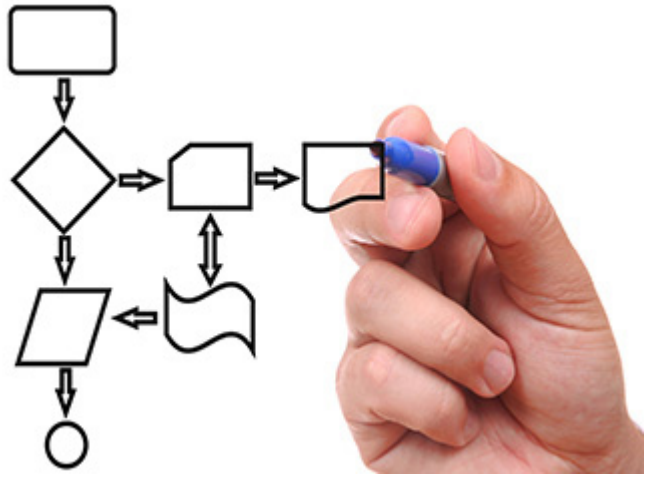
Metodología de programación

Framework usado para estructurar, planear y controlar el **proceso** de desarrollo en sistemas de información.



2. Enfoques

2.1. Framework lineal
2.2. Framework iterativo
2.3. Framework lineal-iterativo
2.4. Otros enfoques



2. Enfoques

2.1. Framework lineal

En sus inicios, la manera de encarar el desarrollo de software **era exclusivamente** lineal o secuencial.

2. Enfoques

2.1. Framework lineal

Modelo de cascada

Proyecto **dividido** en **fases secuenciales**, con cierta superposición aceptable.
Se hace énfasis en la planificación, los horarios, las fechas, el presupuesto y la ejecución de todo un sistema de una sola vez.
Un **estricto control** se mantiene durante la vida del proyecto a través de la utilización de una detallada documentación escrita.

2. Enfoques

2.2. Framework iterativo

Prototipado

Permite desarrollar **modelos de aplicaciones de software** dejando ver su funcionalidad básica, sin necesariamente incluir toda la lógica o características del modelo terminado.
Permite al cliente evaluar el producto en forma temprana e interactuar con los desarrolladores para saber si se está cumpliendo con las expectativas y funcionalidades acordadas.

Tipos de prototipos: homogéneos, heterogéneos, desechables, incrementales

2. Enfoques

2.2. Framework iterativo

RAD

El **Desarrollo Rápido de Aplicaciones** (Rapid Application Development) es una metodología de desarrollo de software que implica **construcción de prototipos** y **desarrollo iterativo**.

- "Comprar puede ahorra dinero en comparación de construir"
- Desarrollo a nivel de abstracción mayor
- Dependencia de terceros.

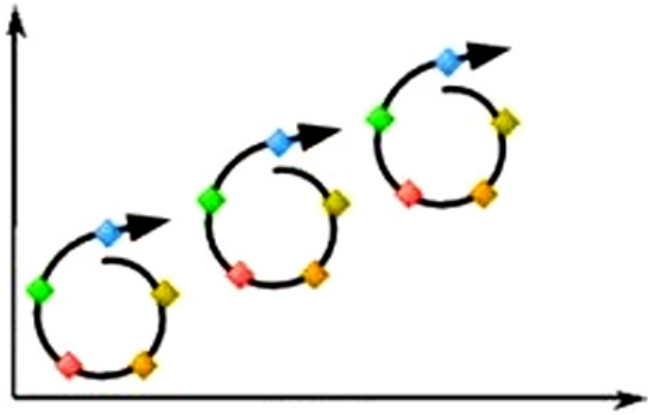
2. Enfoques

2.3. Framework lineal-iterativo

Modelo incremental

Provee una **estrategia** para controlar la complejidad y los riesgos, desarrollando una parte del producto de software reservando el resto de los aspectos para el futuro.

En cada fase se agrega una nueva funcionalidad



2. Enfoques

2.3. Framework lineal-iterativo

Modelo incremental

Principios básicos:

-Una serie de **mini-cascadas** se llevan a cabo, donde todas las fases de la cascada modelo de desarrollo se han completado para una pequeña parte de los sistemas, antes de proceder a la próxima fase.

2. Enfoques

2.3. Framework lineal-iterativo

Modelo incremental

Principios básicos:

-Se definen los **requisitos** antes de proceder con lo evolutivo y se realiza una mini-cascada de desarrollo de cada uno de los incrementos del sistema.

2. Enfoques

2.3. Framework lineal-iterativo

Modelo incremental

Principios básicos:

-El concepto inicial de software, el **análisis** de las necesidades y el **diseño** de la arquitectura se definen **utilizando el enfoque de cascada, seguida por una iteración de prototipos** que culmina en la instalación del prototipo final.

2. Enfoques

2.3. Framework lineal-iterativo

Espiral

La atención se centra en la evaluación y reducción del riesgo, **dividiendo** el proyecto en **segmentos** más pequeños y proporcionando más facilidad de cambio durante el proceso de desarrollo, así como ofreciendo la oportunidad de evaluar los riesgos durante todo el ciclo de vida.

2. Enfoques

2.3. Framework lineal-iterativo

Espiral

Cada vuelta pasa por **4 cuadrantes**:

-Determinación de objetivos, alternativas y desencadenantes de la iteración;

-Evaluación de alternativas e identificación y resolución de riesgos;

-Desarrollo y verificación de los resultados de la iteración

-Planificación de la próxima iteración.

2. Enfoques

2.4. Otros enfoques

Programación top-down

Primero se formula un resumen del sistema, sin especificar detalles.
Cada parte se refina diseñando cada vez con más detalle, hasta que la especificación completa es lo suficientemente detallada para validar el modelo.
Se diseña con frecuencia con la ayuda de "cajas negras".

2. Enfoques

2.4. Otros enfoques

Programación top-down

-Enfatiza la planificación y el conocimiento completo del sistema.
-Se entiende que la codificación no puede comenzar hasta que no se haya alcanzado un nivel de detalle suficiente, al menos en alguna parte del sistema. Esto retrasa las pruebas de las unidades funcionales del sistema hasta que gran parte del diseño se ha completado.

2. Enfoques

2.4. Otros enfoques

Orientado a Objetos

Induce a los programadores a pensar en términos de **objetos** en lugar de **procedimientos**.

Un objeto agrupa datos encapsulados y procedimientos para representar una entidad.

El modelo incluye seis diagramas: **clase**, **objeto**, **estado de transición**, **interacción**, **módulo** y **proceso**.

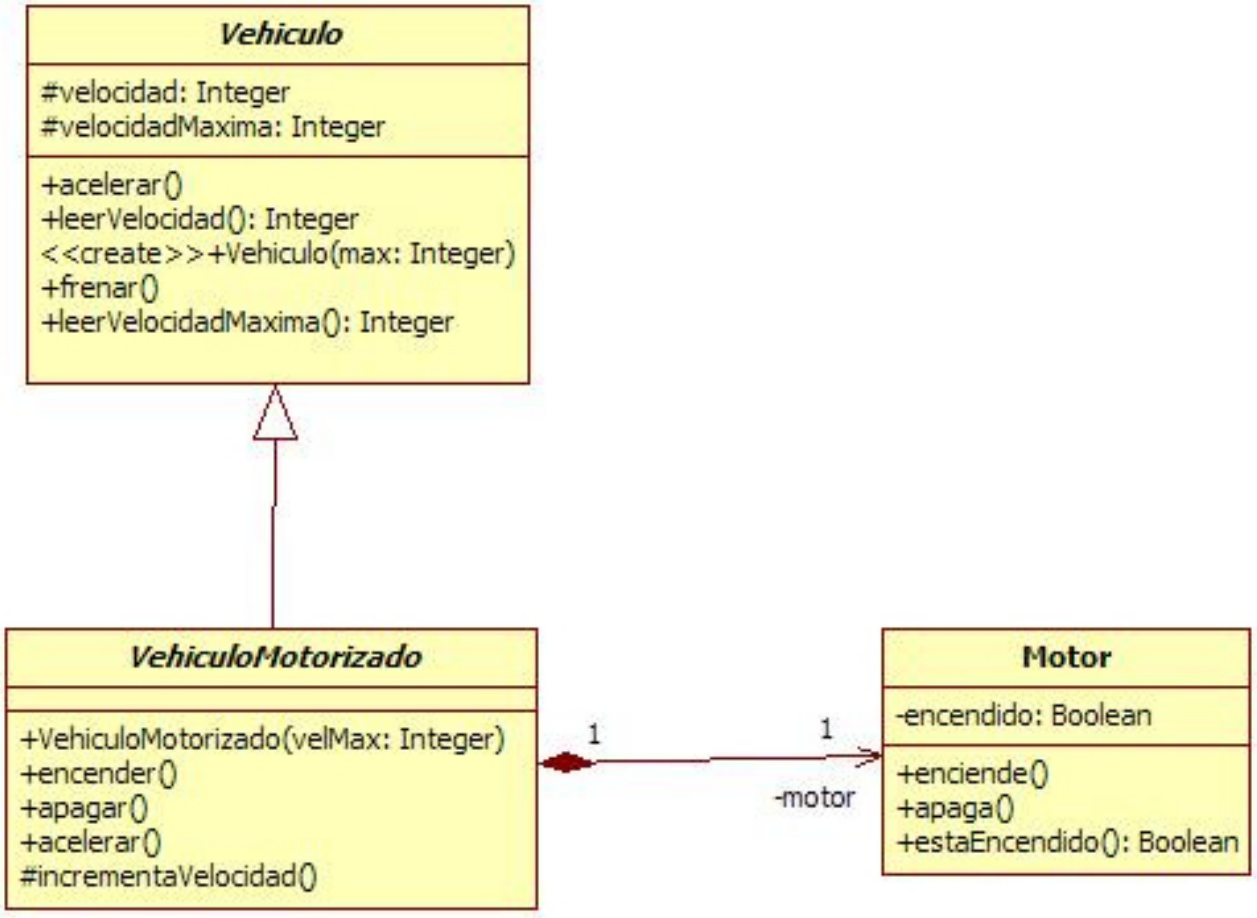
2. Enfoques

2.4. Otros enfoques

Proceso unificado

Metodología de desarrollo de software basada en **UML** (Lenguaje Unificado de Modelado o *Unified Modeling Language*), el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad.

UML


Vehiculo
#velocidad: Integer
#velocidadMaxima: Integer
+acelerar()
+leerVelocidad(): Integer
<<create>>+Vehiculo(max: Integer)
+frenar()
+leerVelocidadMaxima(): Integer
VehiculoMotorizado
+VehiculoMotorizado(velMax: Integer)
+encender()
+apagar()
+acelerar()
#incrementaVelocidad()
1
1
-motor
Motor
-encendido: Boolean
+enciende()
+apaga()
+estaEncendido(): Boolean

2. Enfoques

2.4. Otros enfoques

Proceso unificado

Organiza el desarrollo de software en **cuatro fases**, cada una de ellas con la ejecución de una o más iteraciones de desarrollo de software: **creación**, **elaboración**, **construcción** y directrices.

2. Enfoques

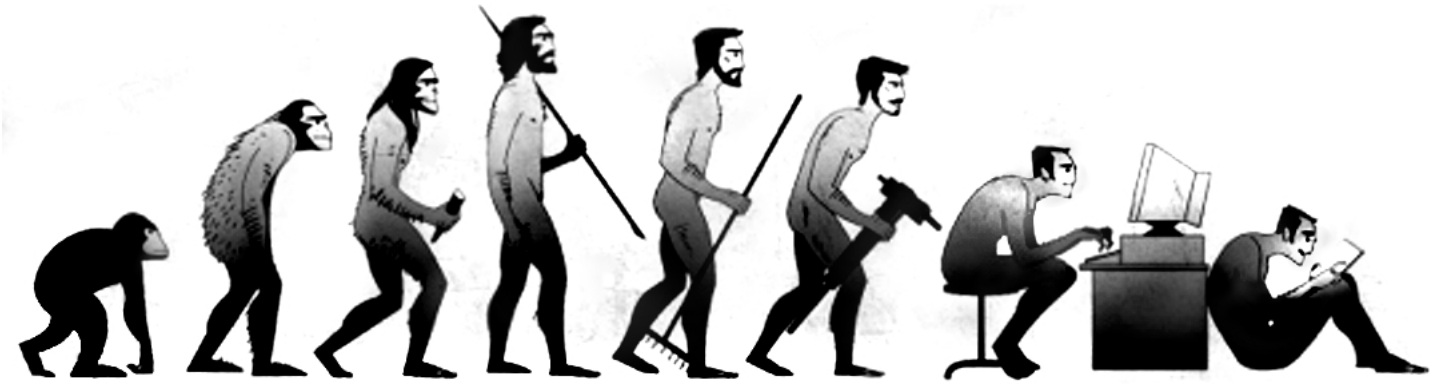
2.4. Otros enfoques

Proceso unificado

Hay una serie de herramientas y productos diseñados para facilitar la aplicación. Una de las versiones más populares es la de **Rational Unified Process**.

3. Historia

3.1. Década de 1960
3.2. Década de 1970
3.3. Década de 1980
3.4. Década de 1990
3.5. Siglo XXI



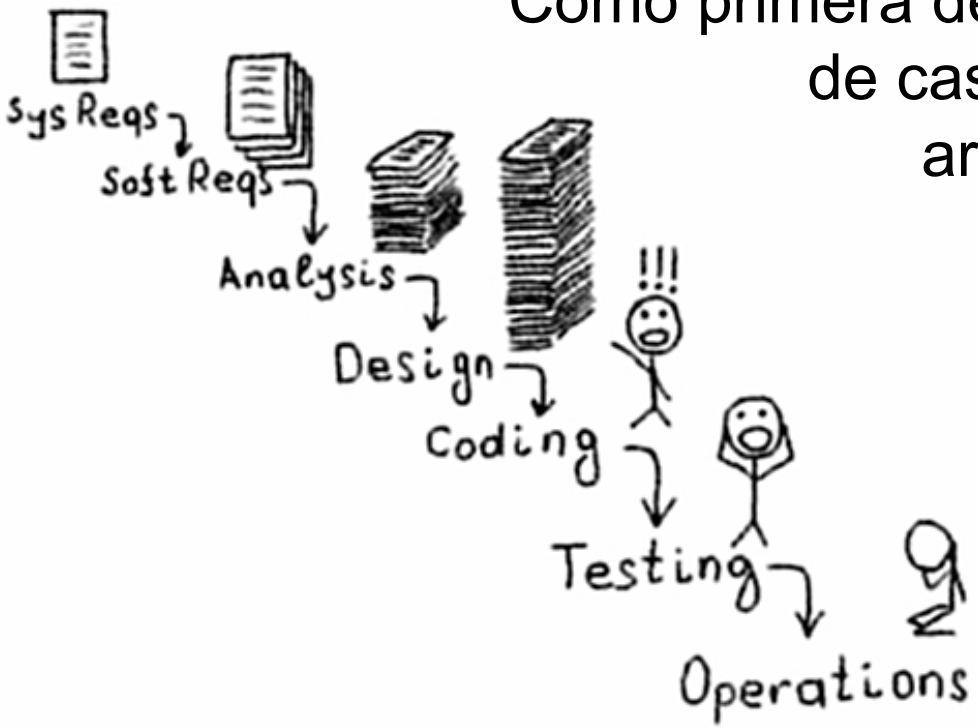
3. Historia

3.1. Década de 1960

El desarrollo de los sistemas tradicionales de **ciclo de vida** se originó en la década de 1960 para producir software para sistemas de negocios en una época de grandes conglomerados empresariales. **Buscaban continuar con el desarrollo de manera estructurada y metódica**

3. Historia

3.2. Década de 1970 (1)



Como primera descripción formal del modelo de cascada se cita a menudo a un artículo publicado por Winston Royce en 1970, aunque no utiliza el término **"CASCADA"**.

Hace hincapié en la planificación, horarios, fechas, presupuestos. Uso de un control estricto.

3. Historia

3.2. Década de 1970 (2)

El diseño Top-down fue promovido en los setenta por los investigadores de IBM Harlan Mills y Niklaus Wirth.
El **éxito** hizo que el enfoque top-down se esparciera por IBM y por el resto de la industria de los computadores.

- De sencilla lectura (programa final en forma de bloques o módulos)
- Es iterativo.
- Trabaja sobre diferentes niveles de abstracción

3. Historia

3.2. Década de 1970 (3)

Niklaus Wirth, que entre sus logros está el desarrollo del lenguaje de programación **Pascal**, escribió el artículo Program Development by Stepwise Refinement, que tuvo mucha influencia en la Ingeniería de Software.

Ley de Wirth: *"el software se vuelve más lento más rápido de lo que el hardware se vuelve más rápido"*

3. Historia

3.3. Década de 1980 (1)

- **1980:** Metodología de Análisis y Diseño de Sistemas Estructurados (SSADM). Resistentes a la pérdida de personal. Sobre análisis y diseño, no sobre la codificación. Método cascada (6 etapas).
- POO (dominante desde mediados de la década).

3. Historia

3.3. Década de 1980 (2)

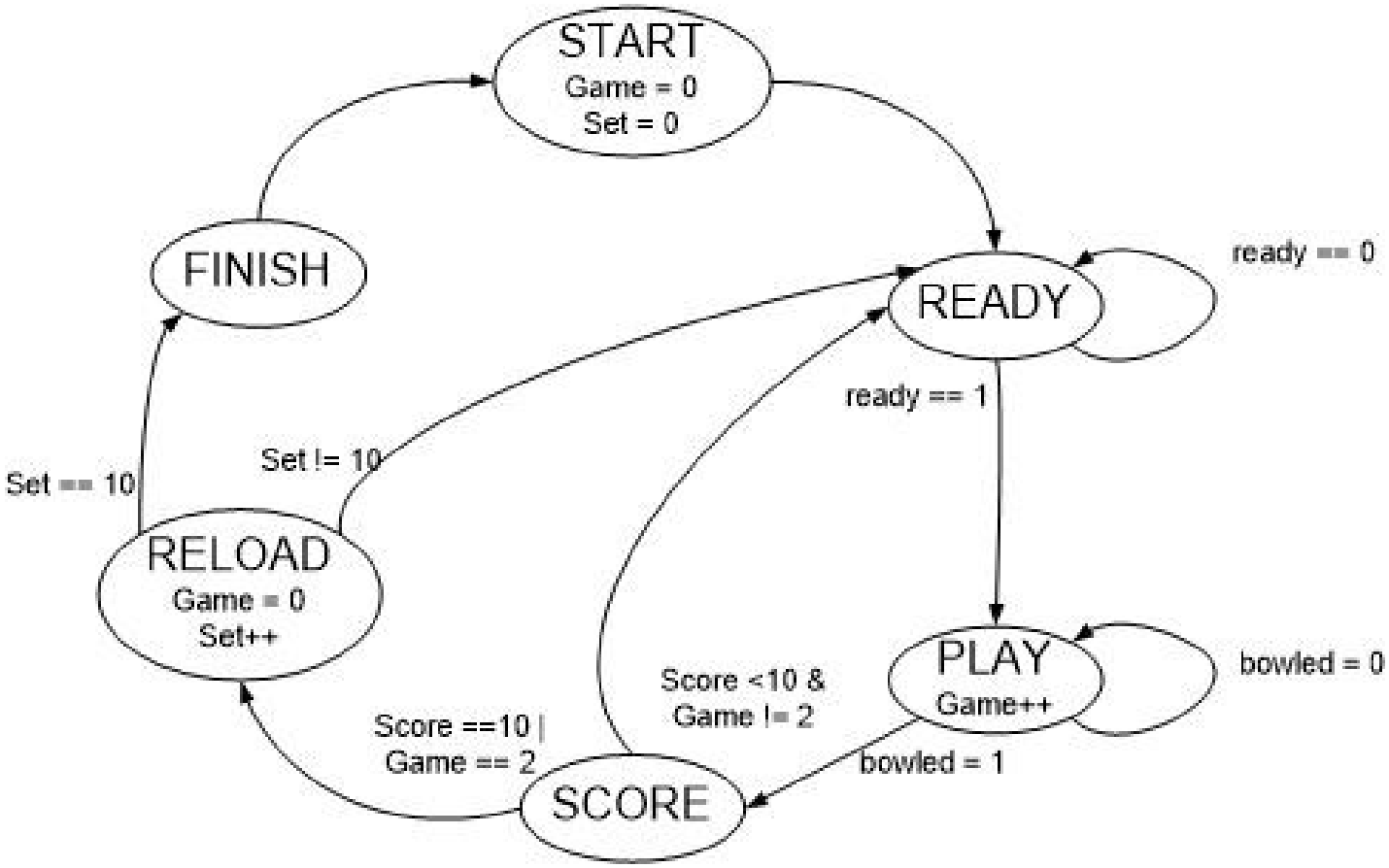
- **1988:** Modelo en espiral, Barry Boehm.
- Primero uso del término SCRUM en sistemas de producción. Aplicación de esta metodología en empresa como: HP, NEC, EPSON, CANON, XEROX, ONDA

3. Historia

3.4. Década de 1990 (1)

- **1991:** Desarrollo Rápido de Aplicaciones(**RAD**). Uso de herramientas CASE, construcción de prototipos y desarrollo iterativo. “Desarrollo a nivel mayor de abstr.”
- **1990:** Virtual finite state machine (VFSM). FSM. Entrada -> Estado -> Salida. Definida en un entorno virtual.

VFSM


START
Game = 0
Set = 0
FINISH
READY
ready == 0
ready == 1
Set != 10
Set == 10
RELOAD
Game = 0
Set++
PLAY
Game++
bowled = 0
Score <10 &
Game != 2
Score ==10 |
Game == 2
bowled = 1
SCORE

3. Historia

3.4. Década de 1990 (2)

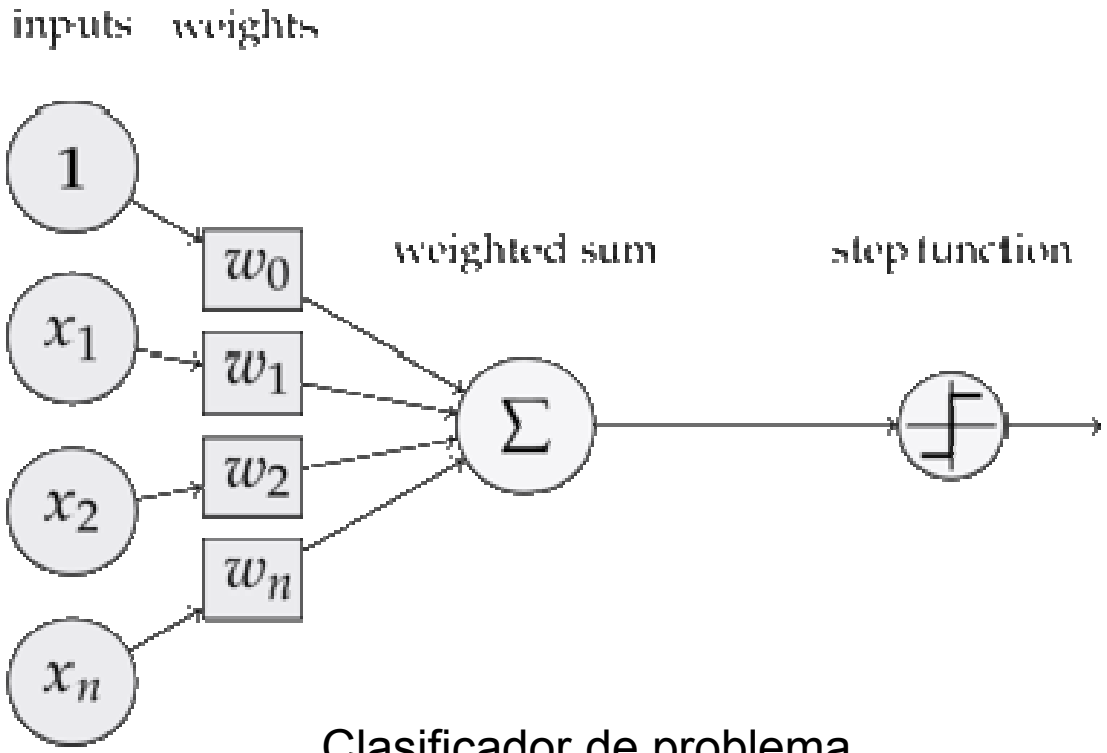
- **1995:** **Scrum** (desarrollo y popularización): publicación de artículo.
- **1995:** "Programación por parejas (idea)
- **1995:** Dynamic Systems Development Method (**DSDM**). Extensión de **RAD** para presupuestos y agendas ajustadas. Todos los cambios son reversibles. El equipo de desarrollo tiene el poder. Pruebas en todas las etapas.
- **1996:** Kent Beck desarrolla (**XP**).
- **1997:** Desarrollo adaptivo de Software (**DAS**)
- **1999:** Rational Unified Process (**RUP**)
- **1999:** Extreme Programming (**XP**). (1er libro)

3. Historia

3.5. Siglo XXI

- **2000:** Alianza SCRUM.
- **2001:** Adopción del nombre “Metodologías Ágiles” y creación de la Alianza Ágil.
- **2004:** Constructionist Design Methodology. De crecimiento iterativo, red de módulos comunicados por mensajes discretos. Relación con la IA.
- **2005:** Agile Unified Process (AUP). Versión simplificada de RUP, enfocada en los riesgos. Los roles pueden ser llevados a cabo por muchas personas y una persona puede tener múltiples roles.

Constructionist Design Methodology (Perceptrón)

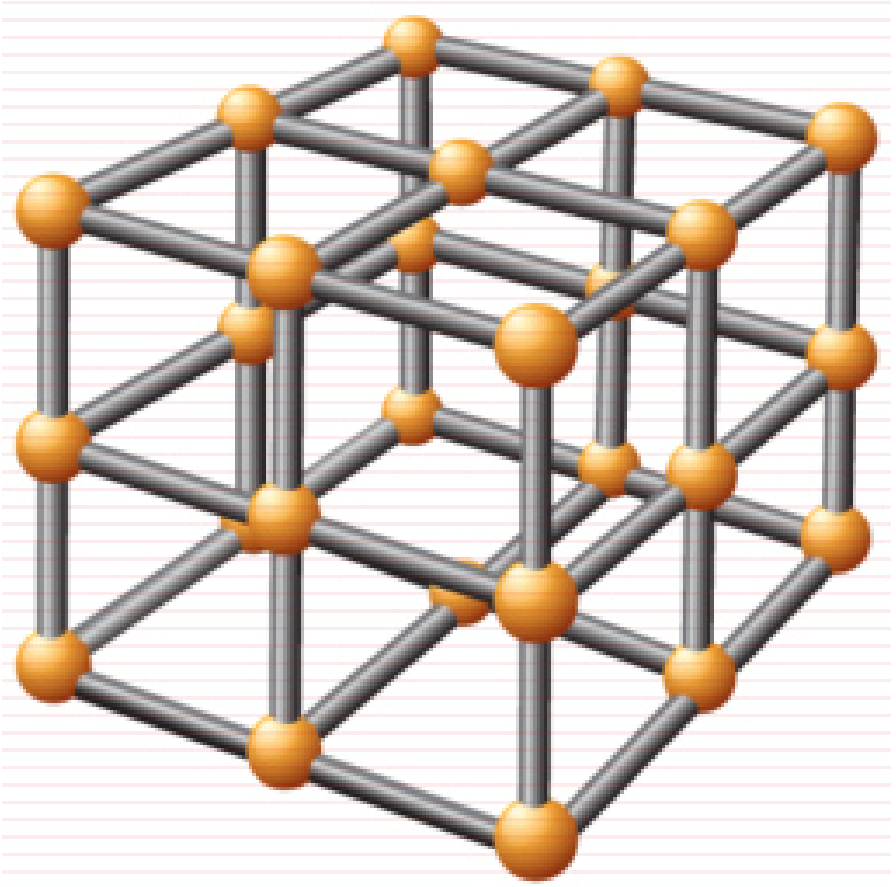


Clasificador de problema

Si hablamos de

desarrollo

del software



¿Qué es un Framework?

4. Framework

4.1 ¿Qué es un Framework?

- Es un **esquema (un esqueleto, un patrón)** para el desarrollo y/o la implementación de una aplicación.

- Es un **entorno o ambiente de trabajo** para el desarrollo de aplicaciones. Dependiendo del lenguaje integra componentes que facilitan el desarrollo de aplicaciones como:
  - el soporte de programas,
  - bibliotecas,
  - plantillas, etc.

4. Framework

4.2 ¿Cuál es su Uso?

- Permite que el trabajo de los desarrolladores sea más **eficiente y recursivo** porque contiene:
  - máquinas virtuales
  - compiladores
  - bibliotecas de administración de recursos en tiempo de ejecución
  - especificaciones de lenguajes

- Su uso depende de las necesidades de cada proyecto y el gusto del desarrollador.

4. Framework

4.2 ¿Cuál es su Uso?

La utilización de un framework en el desarrollo de una aplicación implica:

- un **costo** inicial de **aprendizaje**
- a largo plazo:
  - facilita el **desarrollo**
  - simplifica el **mantenimiento**

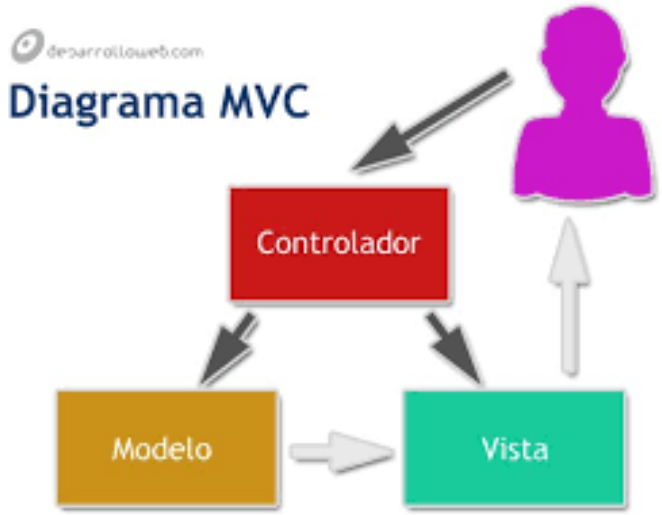
4. Framework

4.3 ¿Cuál es su Arquitectura?

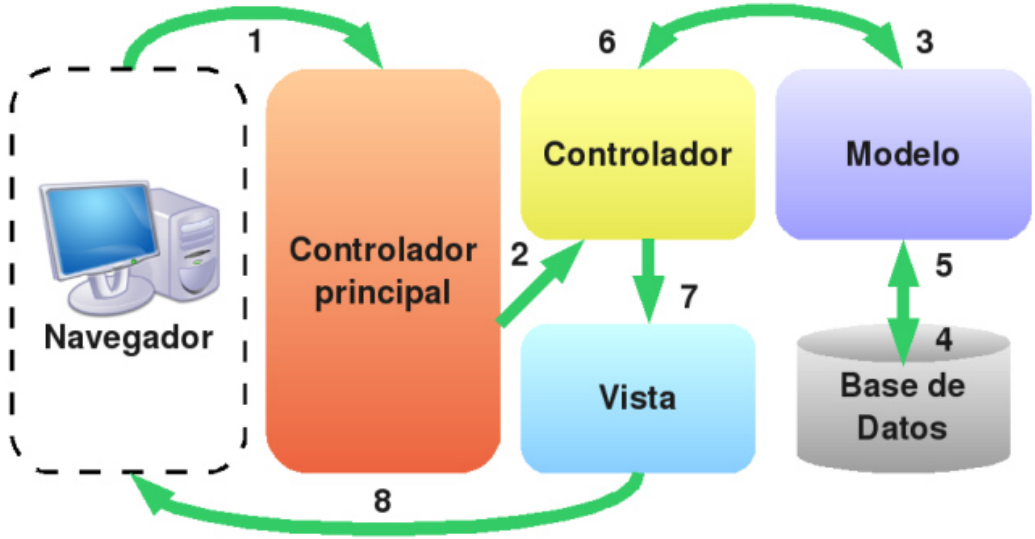
La más utilizada en casi todos los frameworks es conocida como MVC (Modelo, Vista, Controlador).

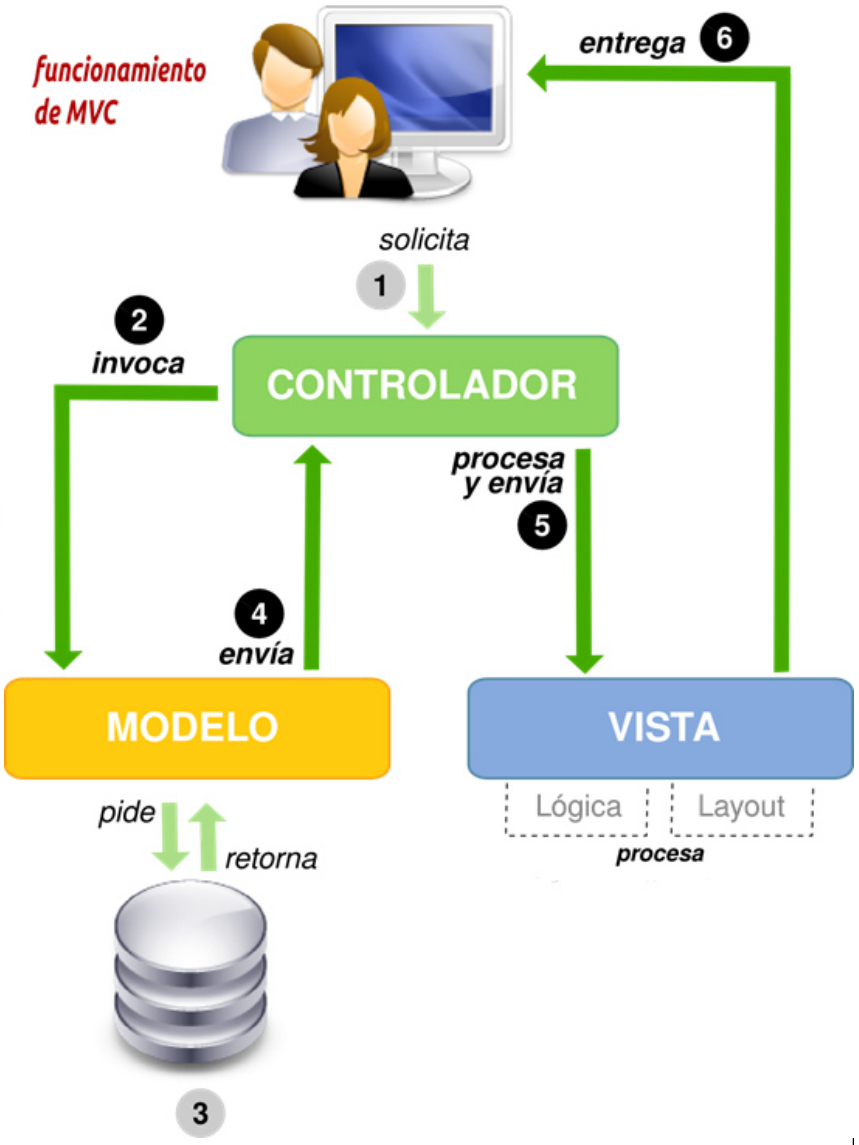
Esta arquitectura divide el desarrollo en tres grandes partes:

- **Modelo:** Son los datos de la aplicación y su reglamentación.
- **Vista:** Es la presentación de los datos.
- **Controlador:** Procesa las peticiones de los usuarios y controla el flujo de ejecución del sistema.



Arquitectura de un Framework





4. Framework

4.4 ¿Cuáles son sus características?

Algunos frameworks comparten las siguientes características:

- La **Autenticación:** mediante login y password permite restringir el acceso y el tipo de permiso.

4. Framework

4.4 ¿Cuáles son sus características?

- El **Acceso a los datos** en archivos txt, xml por ejemplo mediante interfaces que integran la base de datos.

- **Abstracción de URLs y Sesiones** ya que el framework se encarga de manejarlas.

4. Framework

4.4 ¿Cuáles son sus características?

- **Internacionalización:** permite la inclusión de varios idiomas en el desarrollo.

- **Uso de controladores.** Suelen implementar una serie de controladores para la gestión de los eventos y peticiones realizadas a la aplicación.

4. Framework

4.5 ¿Cuáles son sus ventajas?

Las que se derivan de utilizar un estándar:

- el **framework** le proporciona al programador un esqueleto que hay que "rellenar".
- facilita los **desarrollos colaborativos**. Entender y modificar código fuente de otro programador es difícil, por lo tanto, todo lo que sea definir y estandarizar permite ahorrar tiempo y trabajo.
- el desarrollo con framework es menos propenso a sufrir errores.

4. Framework

4.5 ¿Cuáles son sus ventajas?

Otras ventajas relevantes de su uso son:

- Compatibilidad de Lenguajes
- Transparencia de proyectos de plataforma a plataforma
- Portabilidad de Arquitectura
- Integración con múltiples dispositivos.
- Desarrollo de aplicaciones de manera más sencilla, ya que cuenta con los componentes necesarios incluidos.
- Reutilización de Código
- Manejo de Política de diseño uniforme y organizada.

4. Framework

4.6 ¿Y si no necesito o no quiero usar un Framework?

- Un desarrollador puede crear una aplicación sin usar ningún framework conocido.

  Las razones pueden ser:

  - aplicación pequeña
  - falta de conocimiento de un framework que se adapte a sus necesidades,
  - falta de tiempo para seleccionar y utilizar uno.

4. Framework

4.6 ¿Y si no necesito o no quiero usar un Framework?

A medida que la aplicación crece, un programador competente procurará seguir determinadas pautas que le faciliten su trabajo de desarrollo y mantenimiento:

- separación de presentación y lógica,
- una sintaxis coherente, etc.

La evolución natural será construir **su propio framework**.

¿por qué no utilizar uno ya definido, y aprovechar el trabajo de otros muchos desarrolladores?

4. Framework

4.7 ¿Qué Framework utilizo?

- En la red se encuentra mucha información sobre los **frameworks** existentes para las diferentes plataformas y lenguajes.
- La elección del **framework** a utilizar vendrá marcada por:
  - El tipo de aplicación a desarrollar
  - El lenguaje de programación y otras tecnologías concretas: base de datos, sistema operativo, etc.

4. Framework

4.8 ¿Cuántos Frameworks existen?

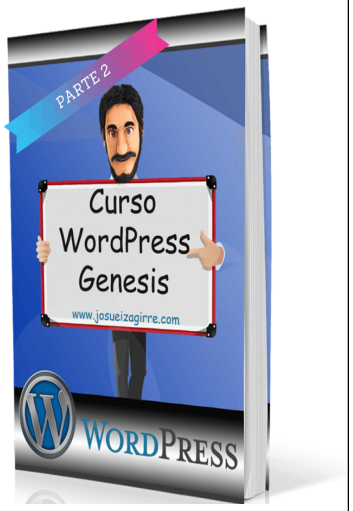
- Existen muchos Frameworks, orientados a diferentes lenguajes, funcionalidades, etc., casi imposible cuantificarlos.

Algunos de ellos son:

- **.NET:** Framework de Microsoft.
- **Ruby on Rails (RoR):** Framework de aplicaciones web de código abierto del lenguaje de programación Ruby.

4. Framework

4.8 ¿Cuántos Frameworks existen?



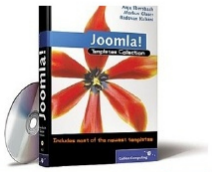
- **Genesis:** Framework para WordPress.

- **Zengrid:** Framework para Joomla.

zengridframework³
Powerful Joomla template framework

Qué es Joomla

- Joomla es un gestor de contenido (CMS), de código abierto escrito en PHP y utiliza como motor de datos mySQL.
- Con Joomla podemos crear portales web, flexibles y altamente escalables
- Joomla es Software Libre!



ELINUX 2011 – PUERTO MONTT 3

4. Framework

4.8 ¿Cuántos Frameworks existen?

- **PhoneGap:** Framework que permite crear aplicaciones móviles multiplataforma a partir de tecnologías web como HTML5, CSS3 y Javascript.

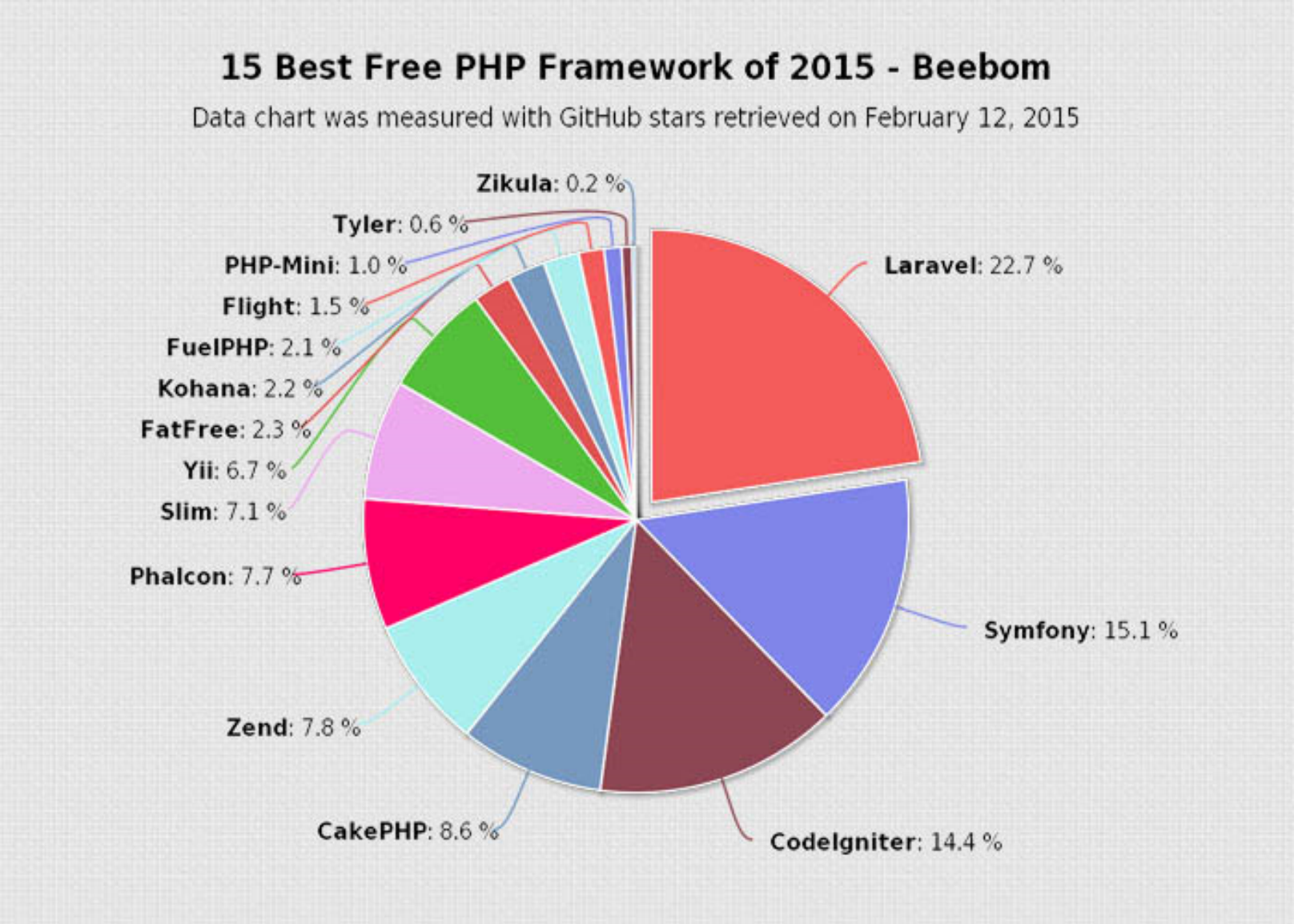
- **Titaniun:** Framework Open Source con licencia Apache, genera aplicaciones nativas para iPhone, Android y Blackberry.

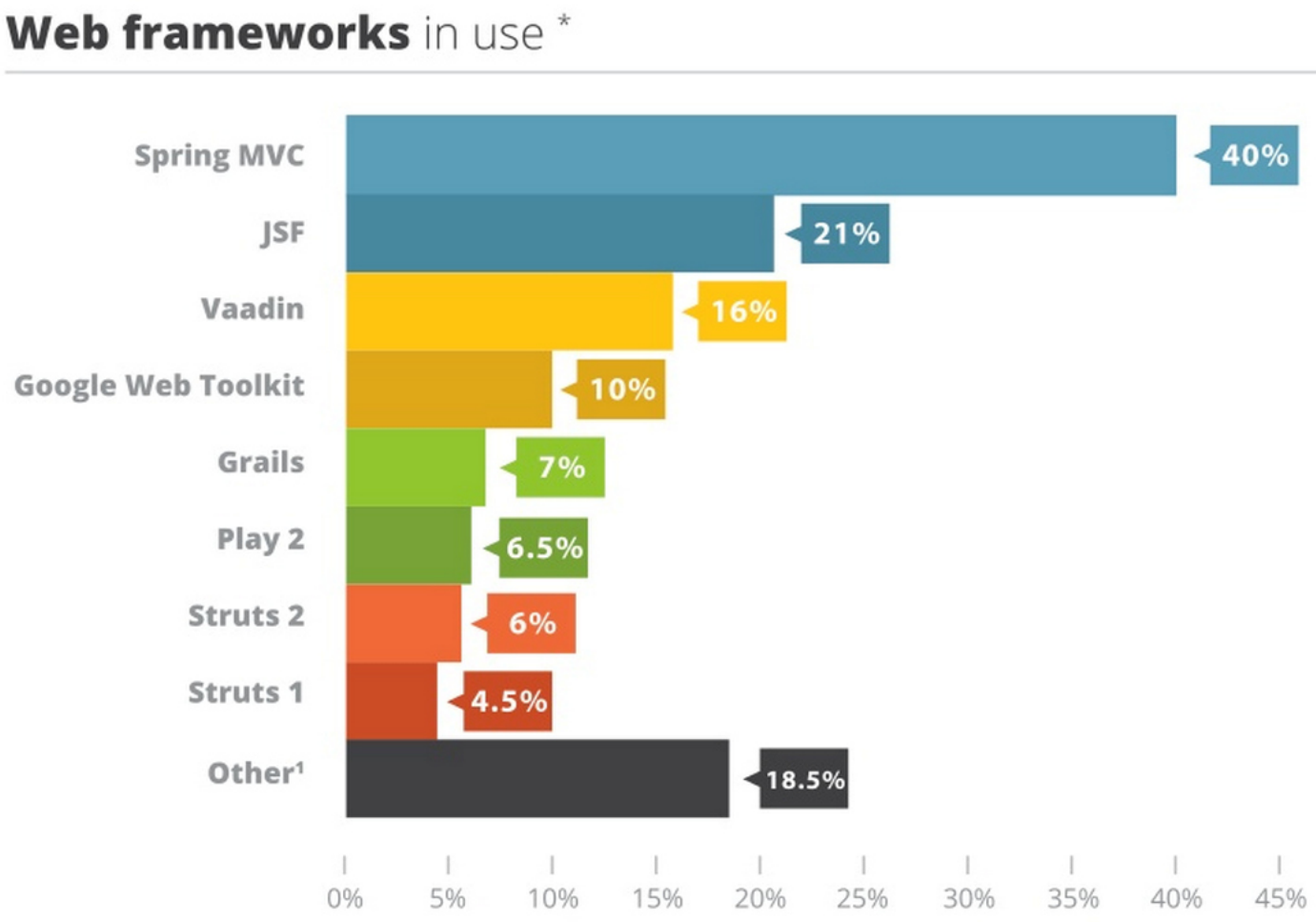
4. Framework

4.8 ¿Cuántos Frameworks existen?

- **JQuery Mobile:** Framework basado en HTML5 optimizado para todas las plataformas de dispositivos móviles.

La lista es interminable, resulta inútil opinar sobre cuál es el mejor framework para utilizar.


15 Best Free PHP Framework of 2015 - Beebom
Data chart was measured with GitHub stars retrieved on February 12, 2015
Zikula: 0.2 %
Tyler: 0.6 %
PHP-Mini: 1.0 %
Flight: 1.5 %
FuelPHP: 2.1 %
Kohana: 2.2 %
FatFree: 2.3 %
Yii: 6.7 %
Slim: 7.1 %
Phalcon: 7.7 %
Zend: 7.8 %
CakePHP: 8.6 %
CodeIgniter: 14.4 %
Symfony: 15.1 %
Laravel: 22.7 %


Web frameworks in use *
Spring MVC
40%
JSF
21%
Vaadin
16%
Google Web Toolkit
10%
Grails
7%
Play 2
6.5%
Struts 2
6%
Struts 1
4.5%
Other[1]
18.5%
0%
5%
10%
15%
20%
25%
30%
35%
40%
45%