FUNDAMENTOS DE PROGRAMACIÓN 25/26

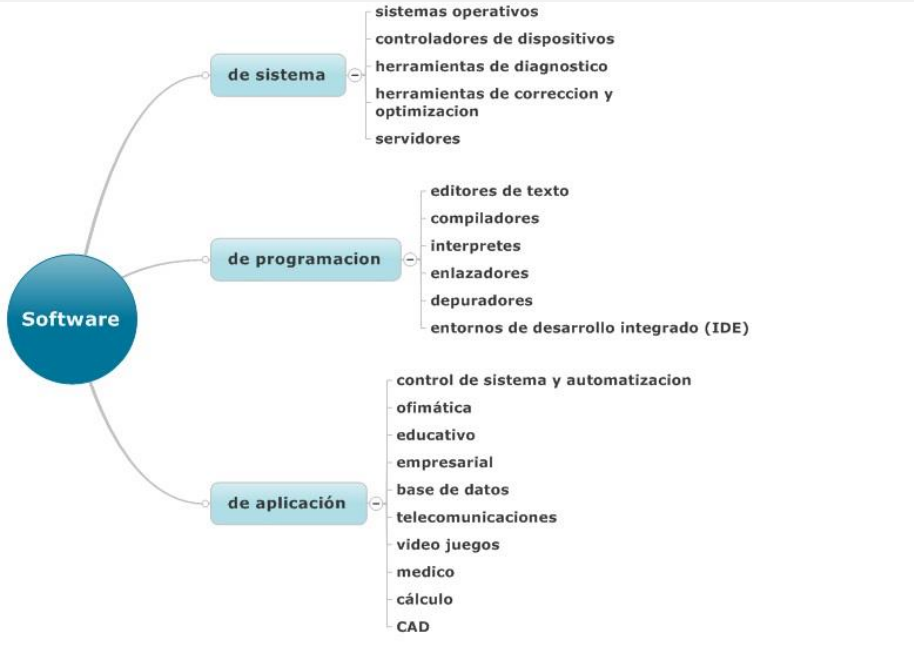
David D´Riu Martínez

INTRODUCCIÓN

- EN ESTA UNIDAD APRENDEREMOS A
- Reconocer la relación de los programas con los componentes del sistema informático.
- Diferenciar código fuente, objeto y ejecutable.
- Identificar las fases de desarrollo de una aplicación informática.
- Clasificar los lenguajes de programación.

TIPOS DE SOFTWARE

- **De sistema** (Sistema operativo, drivers)
- **De aplicación** (Suite ofimática, Navegador, Edición de imagen, ...)
- **De desarrollo** (Editores, compiladores, interpretes, ...)

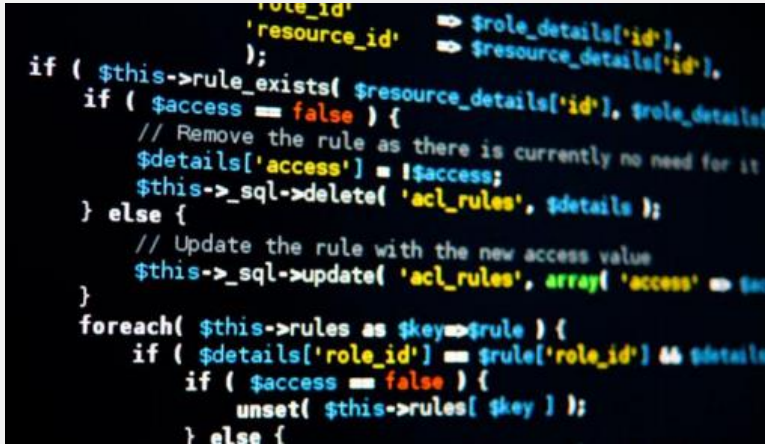


RELACIÓN HARDWARE-SOFTWARE

- **Disco duro**: almacena de forma permanente los archivos ejecutables y los archivos de datos.
- **Memoria RAM**: almacena de forma temporal el código binario de los archivos ejecutables y los archivos de datos necesarios.
- **CPU:** lee y ejecuta instrucciones almacenadas en memoria RAM, así como los datos necesarios.
- **E/S:** recoge nuevos datos desde la entrada, se muestran los resultados, se leen/guardan a disco, ...

CÓDIGOS FUENTE, OBJETO Y EJECUTABLE

- **Código fuente**: archivo de texto legible escrito en un lenguaje de programación
- **Código objeto:** (intermedio): archivo binario no ejecutable.
- **Código ejecutable**: archivo binario ejecutable.



DESARROLLO DE SOFTWARE

- Fases principales:
  - **ANÁLISIS**
  - **DISEÑO**
  - **CODIFICACIÓN**
  - **PRUEBAS**
  - **DOCUMENTACIÓN**
  - **MANTENIMIENTO**

ANÁLISIS

- Se determina y define claramente las necesidades del cliente y se especifica los requisitos que debe cumplir el software a desarrollar. La especificación de requisitos debe:
  - Ser completa y sin omisiones
  - Ser concisa y sin trivialidades
  - Evitar ambigüedades
  - Evitar detalles de diseño o implementación
  - Ser entendible por el cliente
  - Separar requisitos funcionales y no funcionales
  - Dividir y jerarquizar el modelo
  - Fijar criterios de validación

DISEÑO

- Se descompone y organiza el sistema en elementos componentes que pueden ser desarrollados por separado.
- Se especifica la interrelación y funcionalidad de los elementos componentes.
- Las actividades habituales son las siguientes:
  - Diseño arquitectónico
  - Diseño detallado
  - Diseño de datos
  - Diseño de interfaz

CODIFICACIÓN

- Se escribe el código fuente de cada componente.
- Pueden utilizarse distintos lenguajes informáticos:
  - **Lenguajes de programación**: C, C++, Java, Javascript, ...
  - **Lenguajes de otro tipo**: HTML, XML, JSON, ...

PRUEBAS

- El principal objetivo de las pruebas debe ser conseguir que el programa funcione incorrectamente y que se descubran defectos.
- Deberemos someter al programa al máximo número de situaciones diferentes.

DOCUMENTACIÓN

- El principal objetivo de la documentación es explicar como esta realizado el código para que otra persona lo entienda y pueda continuarlo.
- Se especifica lo que realiza para función del programa.

MANTENIMIENTO

- Durante la explotación del sistema software es necesario realizar cambios ocasionales.
- Para ello hay que rehacer parte del trabajo realizado en las fases previas.
- Tipos de mantenimiento:
  - **Correctivo**: se corrigen defectos.
  - **Perfectivo**: se mejora la funcionalidad.
  - **Evolutivo**: se añade funcionalidades nuevas.
  - **Adaptativo**: se adapta a nuevos entornos.

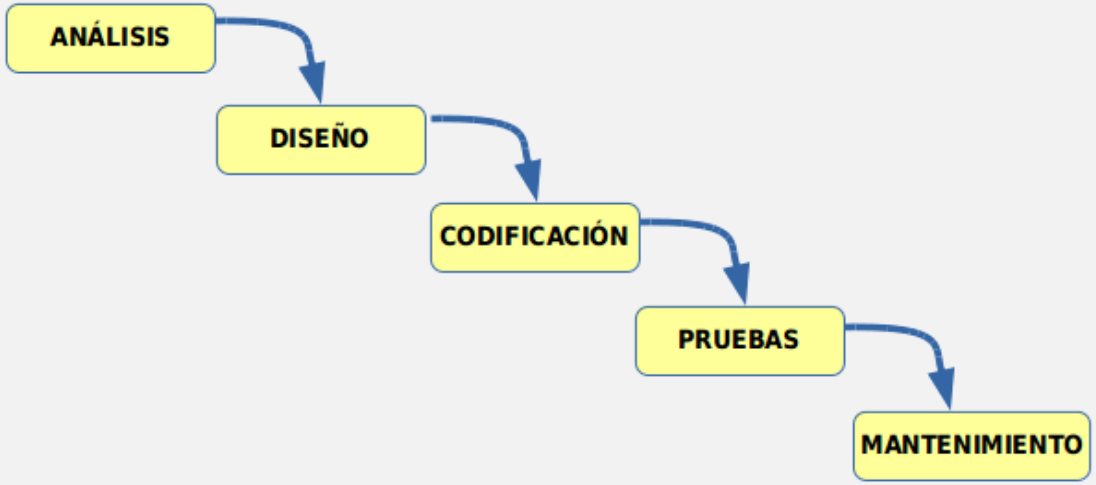
RESULTADO DE CADA FASE

- ANÁLISIS: **Especificación de requisitos del software**
- DISENO arquitectónico: **Documento de arquitectura del software**
- DISENO detallado: **Especificación de módulos y funciones**
- CODIFICACIÓN: **Código fuente**

MODELOS DE DESARROLLO DE SOFTWARE

- Modelos clásicos (predictivos)
  - **Modelo en cascada**
  - **Modelo en V**
- Modelo de construcción de **prototipos**
- Modelos evolutivos o incrementales
  - **Modelo en espiral** (iterativos)
  - **Metodologías ágiles** (adaptativos)

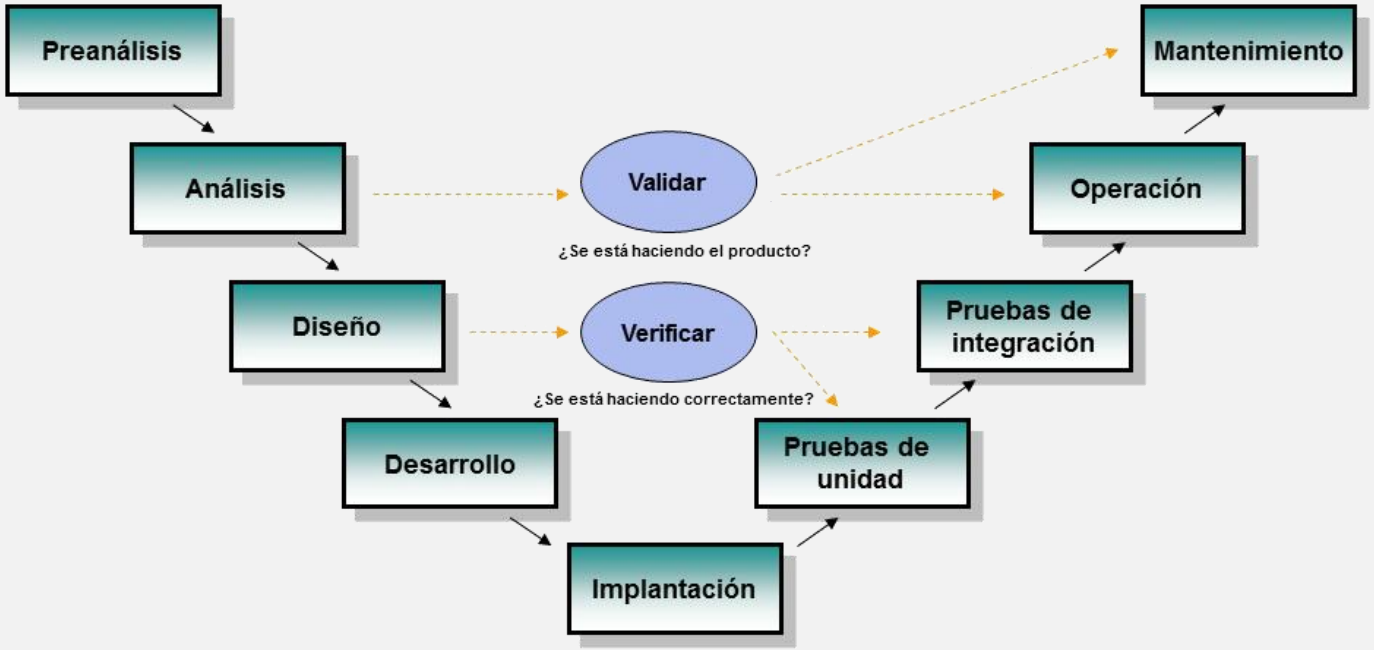
MODELO EN CASCADA



MODELO EN CASCADA

- Modelo de mayor antigüedad.
- Identifica las fases principales del desarrollo software.
- Las fases han de realizarse en el orden indicado.
- El resultado de una fase es la entrada de la siguiente fase.
- Es un modelo bastante rígido que se adapta mal al cambio continuo de especificaciones.
- Existen diferentes variantes con mayor o menor cantidad de actividades.

MODELO EN V

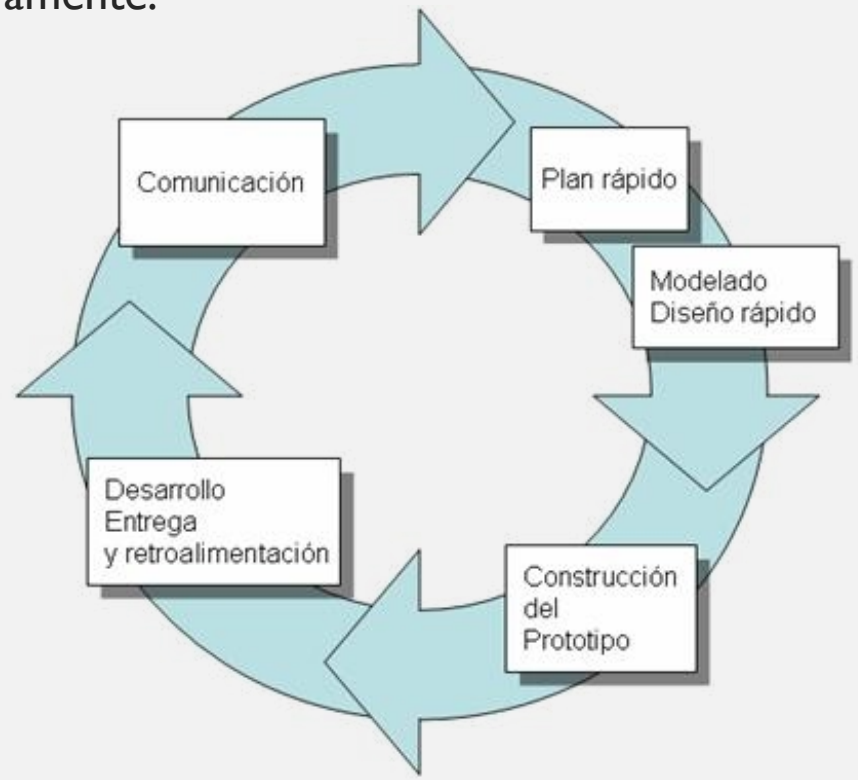

Preanálisis
Análisis
Diseño
Desarrollo
Implantación
Validar
¿Se está haciendo el producto?
Verificar
¿Se está haciendo correctamente?
Pruebas de unidad
Pruebas de integración
Operación
Mantenimiento

MODELO EN V

- Modelo muy parecido al modelo en cascada.
- Visión jerarquizada con distintos niveles.
- Los niveles superiores indican mayor abstracción.
- Los niveles inferiores indican mayor nivel de detalle.
- El resultado de una fase es la entrada de la siguiente fase.
- Existen diferentes variantes con mayor o menor cantidad de actividades.

PROTOTIPOS

- A menudo los requisitos no están especificados claramente:
  - Por no existir experiencia previa.
  - Por omisión o falta de concreción del usuario/cliente.



PROTOTIPOS

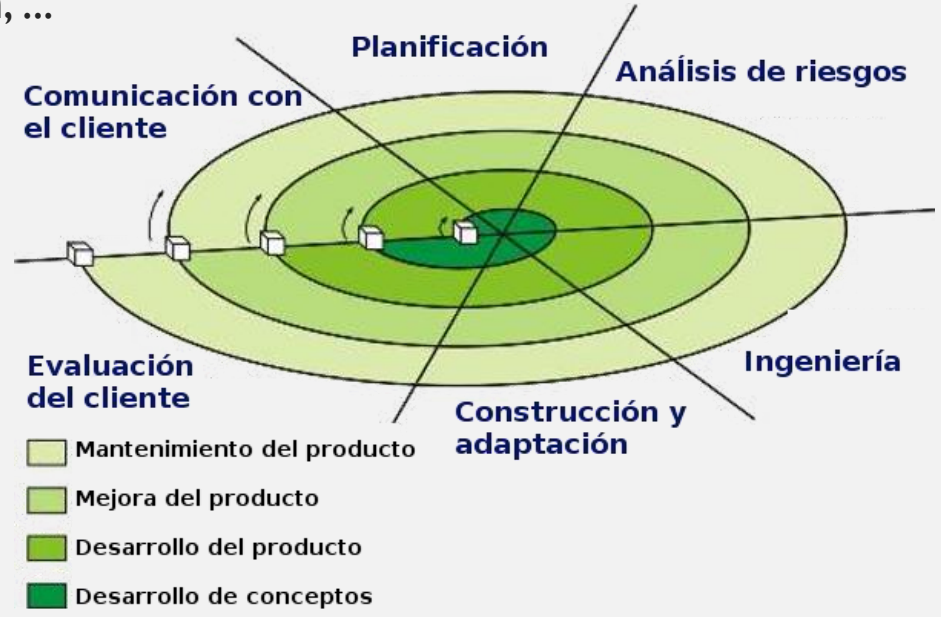
- Proceso:
  - Se crea un prototipo durante la **fase de análisis** y es probado por el usuario/cliente para refinar los requisitos del software a desarrollar.
  - Se repite el paso anterior las veces necesarias.

PROTOTIPOS

- Tipos de prototipos:
  - **Prototipos rápidos**
    - El prototipo puede estar desarrollado usando otro lenguaje y/o herramientas.
    - Finalmente el prototipo se desecha.
  - **Prototipos evolutivos**
    - El prototipo está diseñado en el mismo lenguaje y herramientas del proyecto.
    - El prototipo se usa como base para desarrollar el proyecto.

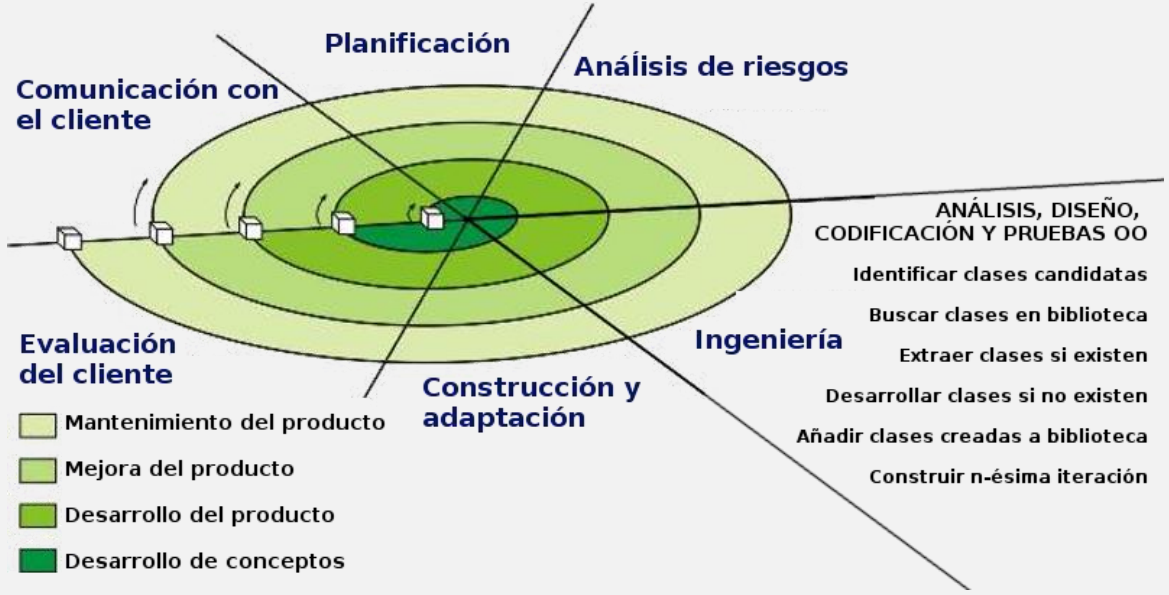
MODELO EN ESPIRAL

- Desarrollado por Boehm en 1988.
- La actividad de ***ingeniería*** corresponde a las fases de los modelos clásicos: análisis, diseño, codificación, ...



MODELO EN ESPIRAL

- ***APLICADO A LA PROGRAMACIÓN ORIENTADA A OBJETOS***
- En la actividad de ***ingeniería*** se da gran importancia a la reutilización de código.



METODOLOGÍAS ÁGILES

- Son métodos de ingeniería del software basados en el desarrollo iterativo e incremental.
- Los requisitos y soluciones evolucionan con el tiempo según la necesidad del proyecto.
- El trabajo es realizado mediante la colaboración de equipos auto-organizados y multidisciplinarios, inmersos en un proceso compartido de toma de decisiones a corto plazo.
- Las metodologías más conocidas son:
  - Kanban
  - Scrum
  - XP (eXtreme Programming)

METODOLOGÍAS ÁGILES

- Son aquellas que permiten adaptar la forma de trabajo a las condiciones del proyecto, consiguiendo flexibilidad e inmediatez en la respuesta para amoldar el proyecto y su desarrollo a las circunstancias específicas del entorno.

METODOLOGÍAS ÁGILES

- Manifiesto por el Desarrollo Ágil

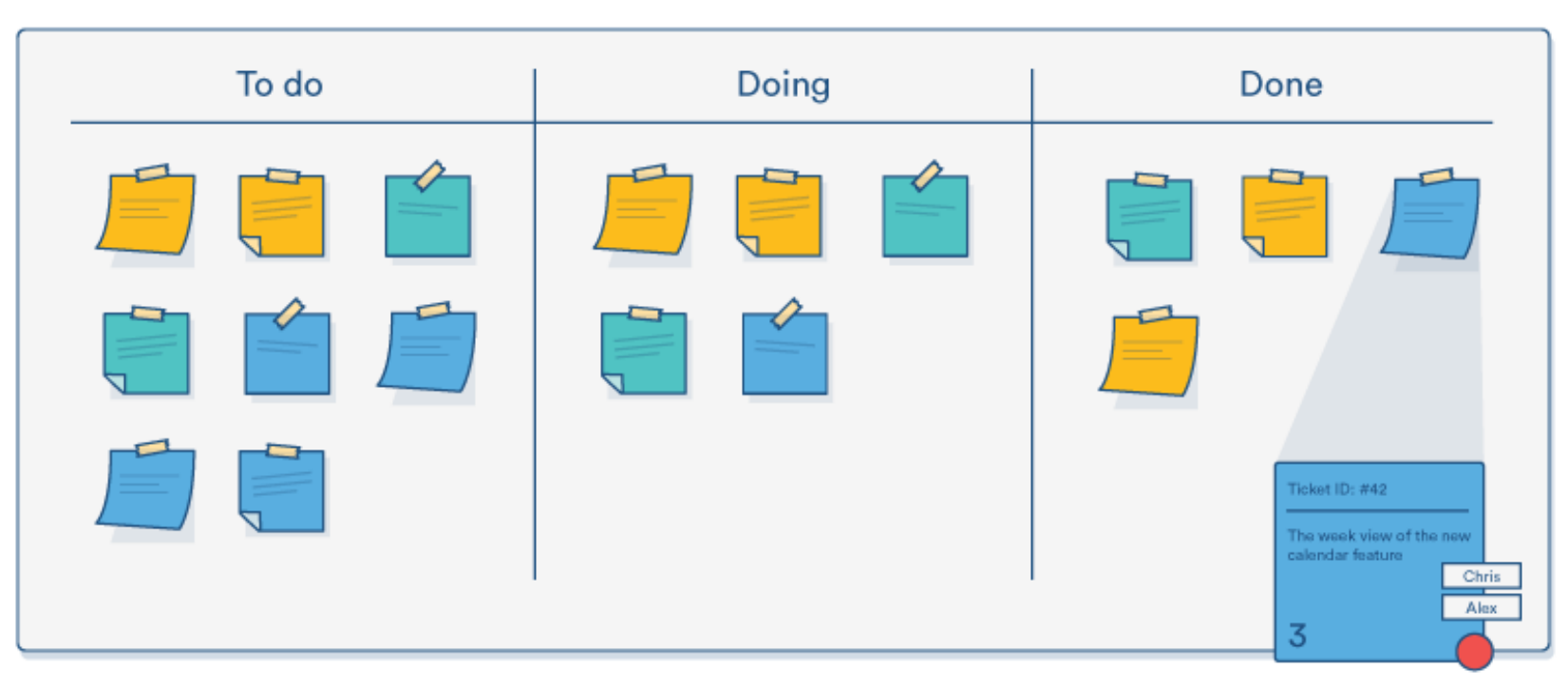
- **Individuos e interacciones** sobre procesos y herramientas
- **Software funcionando** sobre documentación extensiva
- **Colaboración con el cliente** sobre negociación contractual
- **Respuesta ante el cambio** sobre seguir un plan

KANBAN

- También se denomina "sistema de tarjetas".
- **Desarrollado inicialmente por Toyota para la industria de fabricación de productos**.
- **Controla por demanda** la fabricación de los productos necesarios en la cantidad y tiempo necesarios.
- Enfocado a entregar el máximo valor para los clientes, utilizando los recursos justos.
- Lean manufacturing
- Kanban en desarrollo software

KANBAN

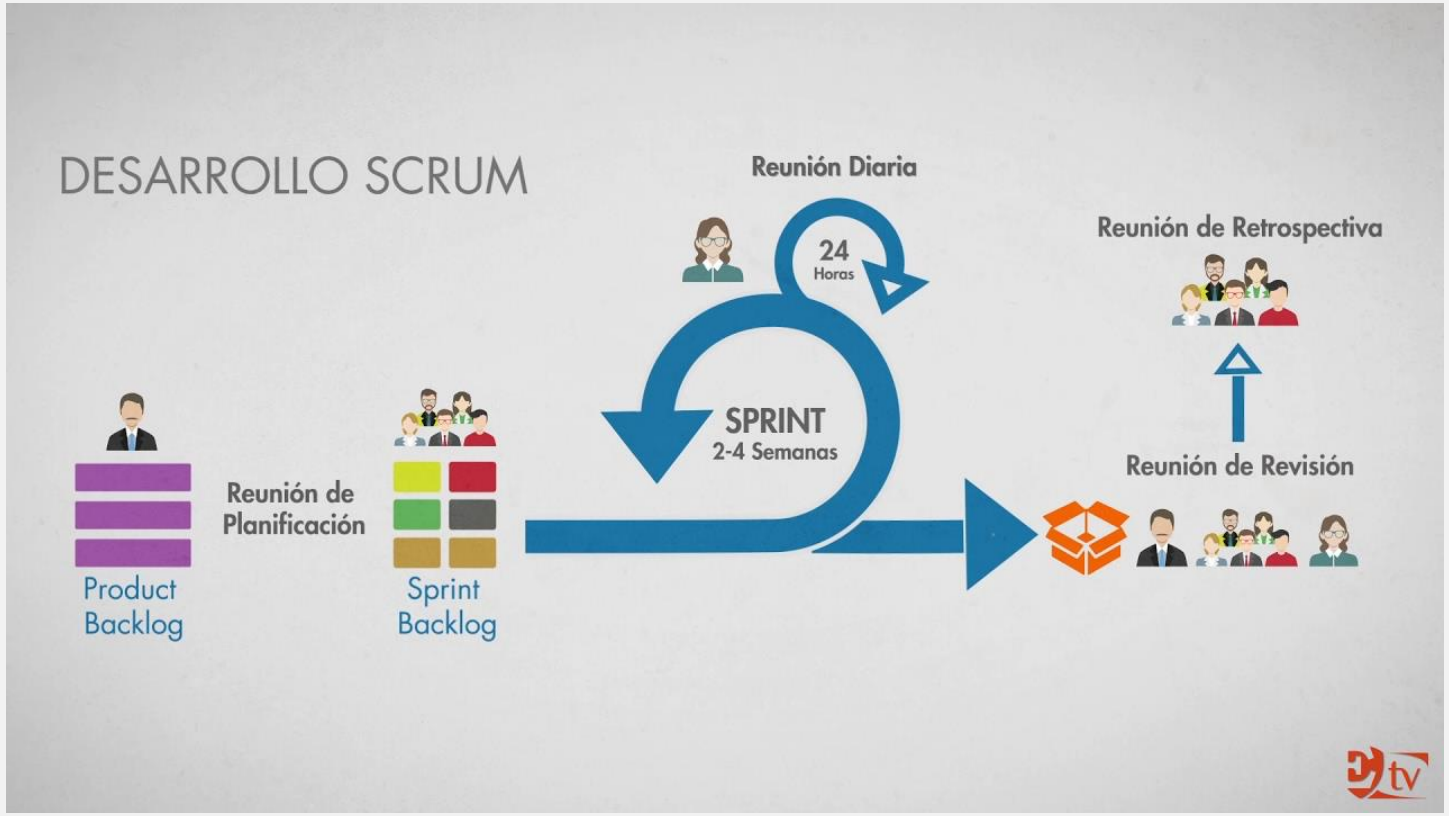
Pizarra Kanban



SCRUM

- Modelo de desarrollo incremental.
- Iteraciones (**sprint**) regulares cada 2 a 4 semanas.
- Al principio de cada iteración se establecen sus **objetivos priorizados** (**sprint backlog**).
- Al finalizar cada iteración se obtiene una **entrega parcial utilizable por el cliente**.
- Existen **reuniones diarias** para tratar la marcha del *sprint*.

SCRUM


DESARROLLO SCRUM
Reunión Diaria
24
Horas
Reunión de Retrospectiva
SPRINT
2-4 Semanas
Reunión de Revisión
Reunión de
Planificación
Product
Backlog
Sprint
Backlog

XP (PROGRAMACIÓN EXTREMA)

- **Valores**
  - Simplicidad
  - Comunicación
  - Retroalimentación
  - Valentía o coraje
  - Respeto o humildad

XP (PROGRAMACIÓN EXTREMA)

- Se basa en:
- Diseño sencillo
- Pequeñas mejoras continuas
- Pruebas y refactorización
- Integración continua
- Programación por parejas
- El cliente se integra en el equipo de desarrollo
- Propiedad del código compartida
- Estándares de codificación
- 40 horas semanales

XP (PROGRAMACIÓN EXTREMA)

- **Roles**
- **Cliente**: responsable de definir y conducir el proyecto así como sus objetivos.
- **Programadores**: estiman tiempos de desarrollo de cada actividad y programan el proyecto.
- **Tester**: Encargado de Pruebas.
- **Tracker**: Encargado de Seguimiento.
- **Coach**: Entrenador. Su papel es guiar y orientar al equipo.
- **Big Boss**: Gestor del proyecto, gerente del proyecto, debe tener una idea general del proyecto y estar familiarizado con su estado.

LENGUAJE DE PROGRAMACIÓN

LENGUAJE DE PROGRAMACIÓN

- Un lenguaje de programación es un conjunto de instrucciones, operadores y reglas de sintaxis y semánticas, que se ponen a disposición del programador para que este pueda comunicarse con dispositivos de hardware y software existentes.

TIPOS DE LENGUAJES DE PROGRAMACIÓN

- **1. Lenguaje máquina:**
- Sus instrucciones son complejas e ininteligibles. Se componen de combinaciones de unos y ceros.
- No necesita ser traducido, por lo tanto es el único lenguaje que entiende directamente el ordenador.
- Fue el primer lenguaje utilizado. En su momento, los expertos debían tener un dominio profundo del hardware para poder entender este lenguaje de programación.
- Difiere para cada procesador. Las instrucciones no son portables de un equipo a otro.
- Salvo excepciones, actualmente, nadie programa en este lenguaje.

TIPOS DE LENGUAJES DE PROGRAMACIÓN

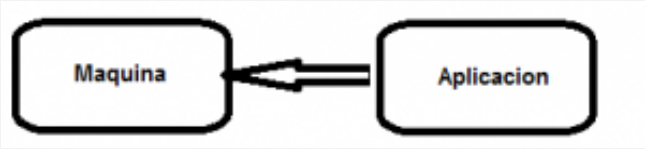
- **2. Lenguaje de medio nivel o ensamblador:**
- Dada la dificultad y poca portabilidad del lenguaje máquina, el ensamblador lo sustituyó para facilitar la labor de programación.
- Sigue estando cercano al hardware, pero, en lugar de unos y ceros, se programa usando mnemotécnicos, que son instrucciones más inteligibles por el programador que permiten comprender de una forma más sencilla qué hace el programa.
- Este lenguaje necesita compilarse y traducirse al lenguaje máquina para poder ejecutarse.
- Se trabaja con los registros del procesador y direcciones físicas. En lenguajes de nivel más alto, ya se utilizan variables y estructuras más sofisticadas.
- Es difícil de comprender y programar.
- Dada la dificultad y poca portabilidad del lenguaje máquina, el ensamblador lo sustituyó para facilitar la labor de programación.

TIPOS DE LENGUAJES DE PROGRAMACIÓN

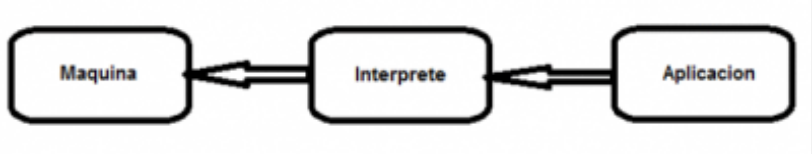
- **3. Lenguajes de alto nivel:**
- La mayoría de los lenguajes de programación actuales pertenecen a esta categoría.
- Tienen una forma de programar más intuitiva y sencilla.
- Más cercano al lenguaje humano que al lenguaje máquina (por ejemplo: WHILE var DO….DONE).
- Suelen tener librerías y funciones predeterminadas que solucionan algunos de los problemas que suelen presentarse al programador.
- En ocasiones, ofrecen Frameworks para una programación más eficiente y rápida.
- Suelen trabajar con mucha abstracción y orientación a objetos. De esa manera, es más fácil la reutilización y el encapsulamiento de componentes.

TIPOS

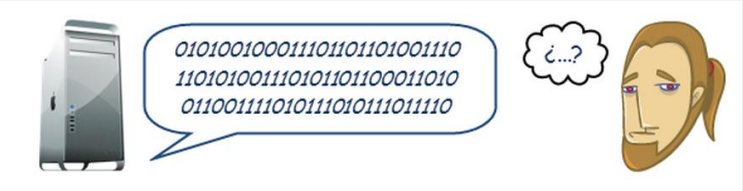
- **Según su forma de ejecución**
- Lenguajes Compilados



- Lenguajes de Nivel Medio



- Lenguajes de Bajo Nivel



TIPOS DE LENGUAJES SEGÚN SU FORMA DE SER EJECUTADOS

Lenguajes compilados.

- Necesitan un programa traductor (compilador) para convertir el código fuente en código máquina. Este tipo de programas se ejecutan de forma más rápida que los interpretados o los virtuales. Además del compilador, existe un programa llamado enlazador o linker que permite unir el código objeto del programa con el código objeto de las librerías.

TIPOS DE LENGUAJES SEGÚN SU FORMA DE SER EJECUTADOS

Lenguajes interpretados.

- No se genera código objeto. El intérprete es un programa que tiene que estar cargado en memoria y se encarga de leer cada una de las instrucciones, interpretarlas y ejecutarlas. Las instrucciones se traducen on the fly y solamente aquellas que van ejecutándose, en vez de interpretar todo el programa.

TIPOS DE LENGUAJES SEGÚN SU FORMA DE SER EJECUTADOS

Lenguajes virtuales.

- Son lenguajes más portables que los lenguajes compilados, puesto que el código que se genera tras la compilación es un código intermedio o Bytecode. Este código puede ser, a su vez, interpretado por una máquina virtual instalada en cualquier equipo. Tienen una ejecución lenta, pero su versatilidad para poder ejecutarse en cualquier entorno los hace muy apreciados.

CARACTERÍSTICAS DE LOS LENGUAJES MAS DIFUNDIDOS

- Java
- Python
- C y C++
- JavaScript
- PHP
- VB.NET

JAVA

- Java se consideró en su momento el lenguaje de internet. Surgió como la evolución de C++ pero adaptado a las nuevas necesidades y tendencias de conectividad.
- Java es una simplificación de C++, dado que no permite la sobrecarga de operadores ni herencia múltiple.
- Para cualquier programador, el manejo de String (cadena de caracteres) en Java es mucho más eficiente y fácil que en C y C++.
- Es un lenguaje orientado a objetos.

JAVA

- Está pensado para trabajar con redes y protocolos TCP/IP, HTTP, FTP, etc.
- Es un lenguaje virtual interpretado.
- Muy portable. Ejecutable en cualquier plataforma.
- Ofrece múltiples aspectos de seguridad. Actualmente, se trabaja en la encriptación del código fuente para una mayor seguridad.
- Permite multihilos. Múltiples hilos de ejecución en un mismo programa.

JAVA

- Ventajas:
  - Estructurado y Orientado a Objetos
  - Relativamente fácil de aprender
  - Buena documentación y comunidad de usuarios
- Desventajas:
  - Menos eficiente que los lenguajes compilados

```
public class HolaMundo {
    public static void main(String[] args) {
    System.out.println("Hola Mundo");
    }
}
```

PHYTON

- Es un lenguaje de alto nivel.
- Su ventaja es la portabilidad que ofrece. Si se evita la dependencia de las librerías particulares de cada sistema, un programa en Python puede ejecutarse en cualquier máquina.
- Es un lenguaje interpretado. Al igual que Java, Python convierte el código fuente en bytecode, que luego se traduce y se ejecuta en la máquina. No hay que enlazar el código con librerías. Eso se hará a la hora de ejecutar el bytecode.

```
# Hola mundo en Python
print ("¡Hola mundo!")
```

PHYTON

- Está orientado a objetos, como la mayoría de lenguajes de programación modernos.
- Permite escribir código en C y luego combinarlo con programas en Python, de esta manera esa porción de código se ejecutará más rápido.
- Puede incrustarse también en otros lenguajes de programación como C y C++.
- Es uno de los lenguajes más simples y sencillos de aprender.

C Y C++

- Son y han sido unos de los lenguajes más potentes y versátiles de la historia de la informática. Mientras que otros lenguajes se han dejado de utilizar, estos siguen utilizándose en múltiples entornos de desarrollo.
- La entrada y salida se ejecuta a través de funciones.
- C estuvo en su momento muy ligado a Unix como sistema operativo, pero C++ intentó desligarse de este.
- Lenguajes estructurados y muy ligados a funciones.

C Y C++

- Incluyen el concepto de puntero, que es una variable que contiene la dirección de memoria de otra variable. Este aspecto ofrece mucha flexibilidad, pero, por otra parte, su utilización por parte del programador puede ser compleja.
- Combinan comandos de alto nivel, aunque pueden incluirse fragmentos de código que trabajen a más bajo nivel.
- Los programas son muy eficientes y rápidos.
- El tamaño de los programas compilados es pequeño.

```
using namespace std;
int main(int argc, char *argv[]) {
    std::cout «"Hola mundo" «endl;
    return 0;
}
```

JAVASCRIPT

- Este lenguaje se utiliza mucho en programación web y sobre todo los Frameworks basados en él como Angular, React, etc.
- Tiene la ventaja de parecerse mucho a Java, C y C++. Por lo tanto, los programadores
- de estos lenguajes se sienten cómodos al programar con JavaScript.
- Es un lenguaje de scripting.
- Se ejecuta en el lado del cliente.
- Es un lenguaje seguro y fiable.

JAVASCRIPT

- Su código es visible y cualquiera puede leerlo, ya que, como se ha explicado, se interpreta en el lado cliente.
- Tiene sus limitaciones como cualquier lenguaje ejecutado en el lado del cliente. Esas
- limitaciones tienen más que ver con el tema de la seguridad.
- Actualmente, existen muchas librerías basadas en JavaScript como AngularJS, ReactJS,
- MeteorJS, JQuery, Foundation JS y BackBone.js, entre otras.

```
<script type="text/javascript">
    alert("Hola Mundo!");
</script>
```

PHP

- Lenguaje web de propósito general utilizado frecuentemente en el backend de muchos productos como PrestaShop, WordPress, etc.
- Lenguaje multiplataforma. Puede instalarse prácticamente en cualquier sistema.
- Se orientó desde el principio al desarrollo de webs dinámicas.
- Conocidos son su buena integración con Mysql y otros servicios como ProFTPd, etc.
- Permite aplicar técnicas de orientación a objetos.
- Lenguaje interpretado, con lo cual no hace falta definir variables.
- Existen muchos Frameworks basados en PHP que permiten trabajar los patrones de diseño modelo-vista-controlador (MVC).

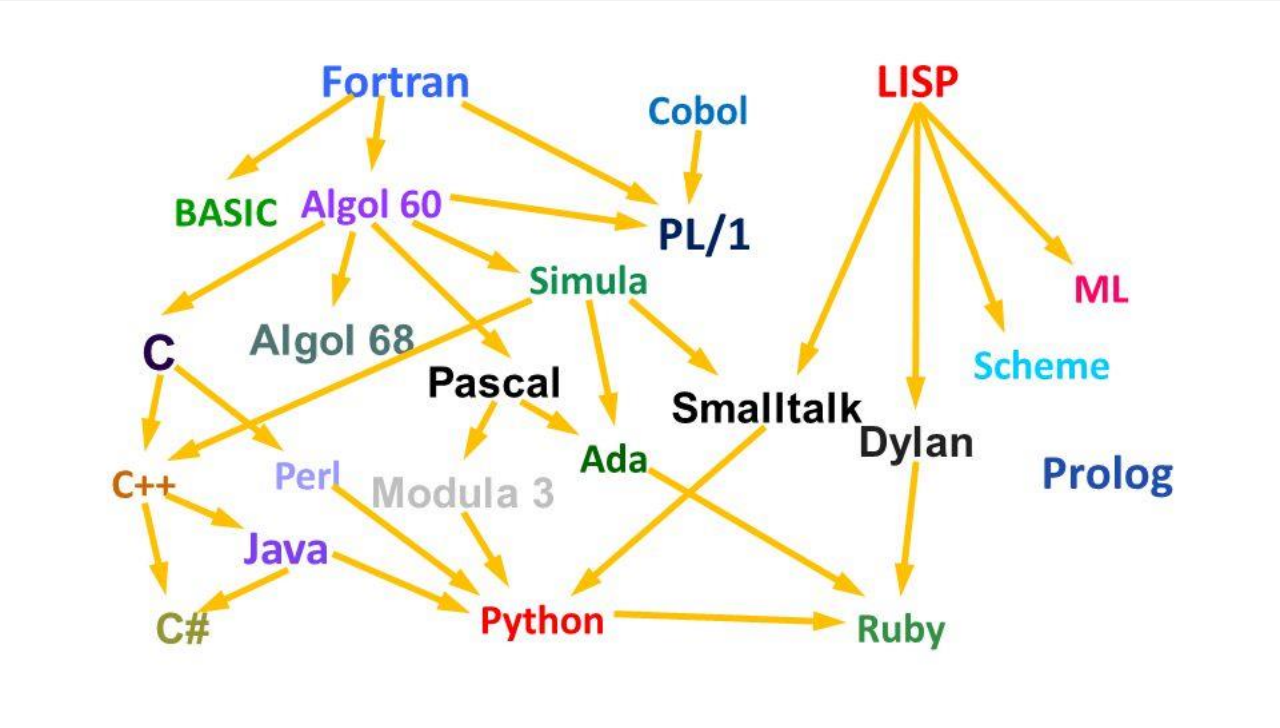
```
<?php echo '<p>Hola Mundo</p>'; ?>
```

VB.NET

- Microsoft tenía que poner al día su famoso Visual Basic y lo consiguió con su nuevo lenguaje VB.NET
- Desde hace mucho tiempo Visual Basic es uno de los referentes de los lenguajes de programación.
- Visual Basic es el lenguaje que se utiliza para programar macros en Microsoft Office.
- VB.NET es un lenguaje orientado a objetos implementado sobre el framework.NET.
- La mayoría de programadores utilizan la herramienta de Microsoft Visual Studio.
- Existen también entornos libres de desarrollo en .NET como Sharpdevelop, pero no es tan potente como el primero.
- Es posible desde el Visual Studio 2008 programar en Ajax.

```
Module VBModule
Sub Main()
Console.WriteLine("Hello, world!")
End Sub
End Module
```

HISTORIA


Fortran
Cobol
LISP
BASIC
Algol 60
PL/1
Simula
ML
C
Algol 68
Scheme
Pascal
Smalltalk
Dylan
Ada
C++
Perl
Modula 3
Prolog
Java
C#
Python
Ruby

CRITERIOS PARA LA SELECCIÓN DE UN LENGUAJE

- Campo de aplicación
- Experiencia previa
- Herramientas de desarrollo
- Documentación disponible
- Reusabilidad
- Portabilidad
- Imposición del cliente