Tema 1

Desarrollo de Software

Entornos de Desarrollo

Objetivos

Reconocer la relación de los programas con los del sistema informático.

Identificar las fases de desarrollo de una aplicación informática

Diferenciar los conceptos de código fuente, código objeto y código ejecutable.

Clasificar los lenguajes de programación.

Conceptos básicos

Un ordenador es un **dispositivo electrónico programable capaz de almacenar y procesar información**. Pero por sí solo, no es capaz de hacerlo, necesita todo un sistema a su alrededor para realizar estas tareas.

Un sistema informático, término utilizado para referirse al conjunto de recursos que son necesarios para la elaboración y el uso de aplicaciones informáticas, está sostenido por los tres elementos básicos siguientes:

- El elemento **físico**, conocido con el nombre de **hardware**
- El elemento **lógico**, conocido con el nombre de **software**.
- El elemento **humano**, conocido como usuario.

Conceptos básicos

El elemento físico (Hardware)

El hardware engloba a todos aquellos elementos con entidad física que forman parte del sistema informático, es decir, son palpables, materiales. Son objetos tales como los componentes del propio ordenador y dispositivos externos (por ejemplo, procesador, discos duros, memorias, la impresora, el teclado, los cables de conexión entre elementos o las unidades donde se guarda la información, etc.).

El ordenador debe almacenar información y procesarla, para ello, se necesitan los elementos capaces de obtener esta información, los que están preparados para guardarla, aquellos dedicados a manipularla y, por último, otros cuya misión sea hacerla llegar al usuario una vez elaborada. Se necesitan mecanismos que aseguren la comunicación entre la persona y la máquina.

Conceptos básicos

El elemento lógico (Software)

El **software** de un sistema informático es el **conjunto de elementos lógicos, programas, datos, información**, etc. que hacen posible el uso y funcionamiento de los ordenadores

Según la wikipedia: *Colección de programas de ordenador y datos que proporcionan las instrucciones a seguir por un ordenador indicándole que hacer y cuando. En otras palabras, el software es una entidad conceptual que incluye programas, procedimientos algoritmos y su documentación.*

Se puede decir que los elementos básicos del software son los datos y las órdenes o instrucciones. Si el software forma parte del sistema informático, deberá almacenarse en un soporte físico como la memoria central o la memoria secundaria.

Conceptos básicos

El elemento lógico (Software)

Software	Sistema Operativo		Windows 7, Windows Vista... Ubuntu, RedHat, Solaris... Mac OS X Lion, Mac OS X Leopard...
	Aplicaciones	Ofimática	Microsoft Office...
		Programas de dibujo	Gimp, Photoshop, Autocad...
		Reproductores de música	Media Player, VLC, Winamp...
		Juegos	Need for speed,
		...	...

Conceptos básicos

El elemento lógico (Software)

Sistema Operativo es la herramienta lógica del sistema informático que controla el funcionamiento del equipo físico y gestiona todos los recursos haciendo transparente al usuario las características físicas de la máquina, facilitando de este modo su uso y mejorando su eficacia.

Son ejemplos de sistemas operativos, MS-DOS, UNIX, Linux, OS/2, OS-400, Windows XP, Mac OS-X

Conceptos básicos

El elemento lógico (Software)

El **Software de Aplicación** está formado por un conjunto de programas diseñados con el objetivo de que los ordenadores realicen trabajos específicos, facilitando al usuario la realización de sus actividades.
Son aplicaciones tales como herramientas ofimáticas (Microsoft Office, OpenOffice), programas de dibujo (CorelDraw, Microsoft Visio), programas para la realización de nóminas, o para llevar la contabilidad de la empresa (Contaplus, Contawin).
Pertenecen también a este grupo de software de aplicación las herramientas de programación para los distintos lenguajes, necesarias para la realización de programas.

Conceptos básicos

Conceptos básicos

El elemento humano

Se llama **usuario a este grupo de personas que utilizan los ordenadores** en última instancia, usando programas de utilidades más o menos complejos creados por otras personas, con el objetivo de ayudarse en alguna actividad.

Y se conoce como **personal informático al conjunto de personas que trabajan para garantizar el correcto funcionamiento de los sistemas de información**, es decir, además de utilizados como herramienta, son el objeto de su trabajo.

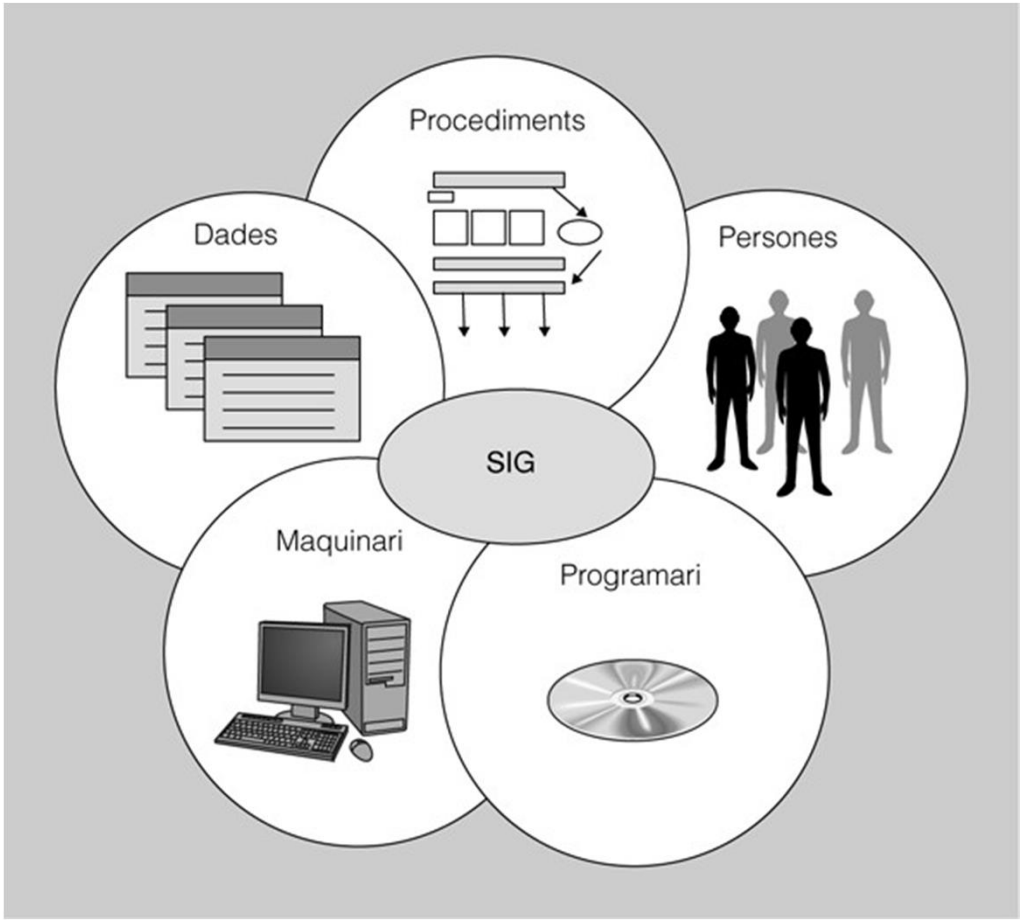
Conceptos básicos

Sistema de información

Se llama Sistema de Información (SI) a una combinación de tecnología, personas y procesos que se utiliza para la gestión de una organización. Es decir es el sistema que dentro de una organización proporciona la información necesaria en cada departamento.
Para respaldar el proceso se utilizan herramientas como ordenadores, dispositivos de comunicación móviles, cámaras... y también software con aplicaciones específicas en cada área.

Conceptos básicos

Sistema de información



Software según en el tipo de tarea que realiza

En terminología informática el **software de sistema**, denominado también **software de base**, consiste en un software que sirve para controlar e interactuar con el sistema operativo, proporcionando control sobre el hardware y dando soporte a otros programas.

Los tipos básicos del software del sistema son:

- La BIOS de la computadora y el firmware del dispositivo, que proporcionan la funcionalidad básica para operar y controlar el hardware conectado o integrado en el equipo.
- El sistema operativo (explicado anteriormente).
- Software de utilidad, lo que ayuda a analizar, configurar, optimizar y mantener el equipo.

Software según en el tipo de tarea que realiza

Software según en el tipo de tarea que realiza

El **Software de Aplicación** son los programas diseñados para o por los usuarios para facilitar la realización de tareas específicas en la computadora, como pueden ser las aplicaciones ofimáticas (procesador de texto, hoja de cálculo, programa de presentación, sistema de gestión de base de datos...), u otros tipos de software especializados como software médico, software educativo, editores de música, programas de contabilidad, etc.

Software según en el tipo de tarea que realiza

Software según en el tipo de tarea que realiza

Como su nombre lo indica, **un software de desarrollo** es un programa que permite el desarrollo de aplicaciones en lenguajes como java, visual basic, c++, Python, Lisp, etc.

El software de desarrollo comúnmente se conoce por IDE (Integrated Development Environment, por sus siglas en inglés). Normalmente cuenta con una avanzada interfaz gráfica de usuario (GUI).

Licencias de software y modelos de negocio

Licencias de software

Una licencia de software es un contrato entre el desarrollador y el usuario del programa informático para utilizar el software cumpliendo una serie de términos y condiciones establecidas dentro de sus cláusulas.

Las licencias de software pueden establecer entre otras cosas: la **cesión de determinados derechos** del desarrollador al usuario final sobre una o varias copias del programa informático, los límites en la **responsabilidad por fallos**, el plazo de cesión de los derechos, el ámbito geográfico de validez del contrato e incluso pueden establecer determinados compromisos del usuario hacia el propietario, tales como la no cesión del programa a terceros o la no reinstalación del programa en equipos distintos al que se instaló originalmente.

Licencias de software y modelos de negocio

Licencias de software

A continuación se muestra una clasificación de los principales tipos de licencias según los derechos que se ceden a los usuarios:

- **Software propietario, código cerrado o privativo**: se permite al usuario utilizar únicamente el programa pero normalmente se restringe su copia o modificación y únicamente se distribuyen los ejecutables. *Simplificando un poco, es el que se puede instalar en una máquina y usar pero no lo puedes pasar a los amigos ni modificar ya que además no tienes el código fuente (cómo está hecho el programa).*
- **Shareware**: suelen ser versiones de software propietario que se distribuyen para prueba por posibles compradores. Pueden estar limitadas en tiempo de uso o capacidades.

Licencias de software y modelos de negocio

Código libre (FreeSoftware): simplificándolo un poco podemos decir que en este caso se permite su uso, modificación y distribución. Dentro exiten dos variantes: una en la que se obliga a que los programas derivados de uno libre también tienen que ser libres, y otros en que a las obras derivadas se les puede cambiar la licencia.

- El software libre aporta las siguientes libertades:
  - Libertad para ejecutar el programa en cualquier sitio, con cualquier propósito y para siempre.
  - Libertad para estudiarlo y adaptarlo a nuestras necesidades. Esto exige el acceso al código fuente.
  - Libertad de redistribución, de modo que se nos permita colaborar con vecinos y amigos.
  - Libertad para mejorar el programa y publicar las mejoras. También exige el código fuente.

Licencias de software y modelos de negocio

Modelos de negocio

Venta de producto cerrado: El desarrollador vende un producto que se supone que no necesita ninguna modificación. Ejemplos pueden ser Microsoft Office, Photoshop, Windows 7... El precio suele ir asociado al número de licencias de usuario.

Shareware: Suele consistir en versiones de prueba de software de producto cerrado. Pero permite que los usuarios prueben la versión antes de comprarlo.

Venta con de servicios de adaptación: Para poder utilizar algunos programas necesitan ser adaptados y configurados a las condiciones del usuario. Esto típicamente sucede con empresas, cuando los programas de gestión tienen que adaptarse al específico de gestión esa empresa. Por ejemplo es común en ERP (Enterprise Resouce Planning) como SAP y SAGE, donde personal de la empresa desarrolladora *implanta* el sistema en la empresa cliente.

Ciclo de vida del Software

El término **ciclo de vida del software** describe el desarrollo de software, desde la fase inicial hasta la fase final. El propósito de este es definir las distintas fases intermedias que se requieren para **validar** el desarrollo de la aplicación, es decir, para garantizar que el software cumpla los requisitos para la aplicación y **verificación** de los procedimientos de desarrollo: se asegura de que los métodos utilizados son apropiados.

Son una serie de etapas o fases que hay que seguir secuencialmente.

Las fases o etapas son:

- Análisis.
- Diseño.
- Codificación.
- Pruebas.
- Mantenimiento.

Ciclo de vida del Software

Análisis de los requisitos del software: el proceso de recopilación de los requisitos se centra e intensifica especialmente en el software. El ingeniero de software (Analistas) debe comprender el ámbito de la información del software, así como la función, el rendimiento y las interfaces requeridas.

Diseño: el diseño del software se enfoca en cuatro atributos distintos del programa: la estructura de los datos, la arquitectura del software, el detalle procedimental y la caracterización de la interfaz. El proceso de diseño traduce los requisitos en una representación del software con la calidad requerida antes de que comience la codificación.

Ciclo de vida del Software

Codificación: el diseño debe traducirse en una forma legible para la maquina. El paso de codificación realiza esta tarea. Si el diseño se realiza de una manera detallada la codificación puede realizarse mecánicamente.

Prueba: una vez que se ha generado el código comienza la prueba del programa. La prueba se centra en la lógica interna del software, y en las funciones externas, realizando pruebas que aseguren que la entrada definida produce los resultados que realmente se requieren.

Mantenimiento: el software sufrirá cambios después de que se entrega al cliente. Los cambios ocurrirán debido a que hayan encontrado errores, a que el software deba adaptarse a cambios del entorno externo (sistema operativo o dispositivos periféricos), o debido a que el cliente requiera ampliaciones funcionales o del rendimiento.

Ciclo de vida del Software

Cada etapa tiene como entrada uno o varios documentos procedentes de la etapa anterior y produce otros documentos de salida, por ello dentro de todas las etapas una de las tareas más importante es la "**Documentación**".

Modelos de ciclo de vida

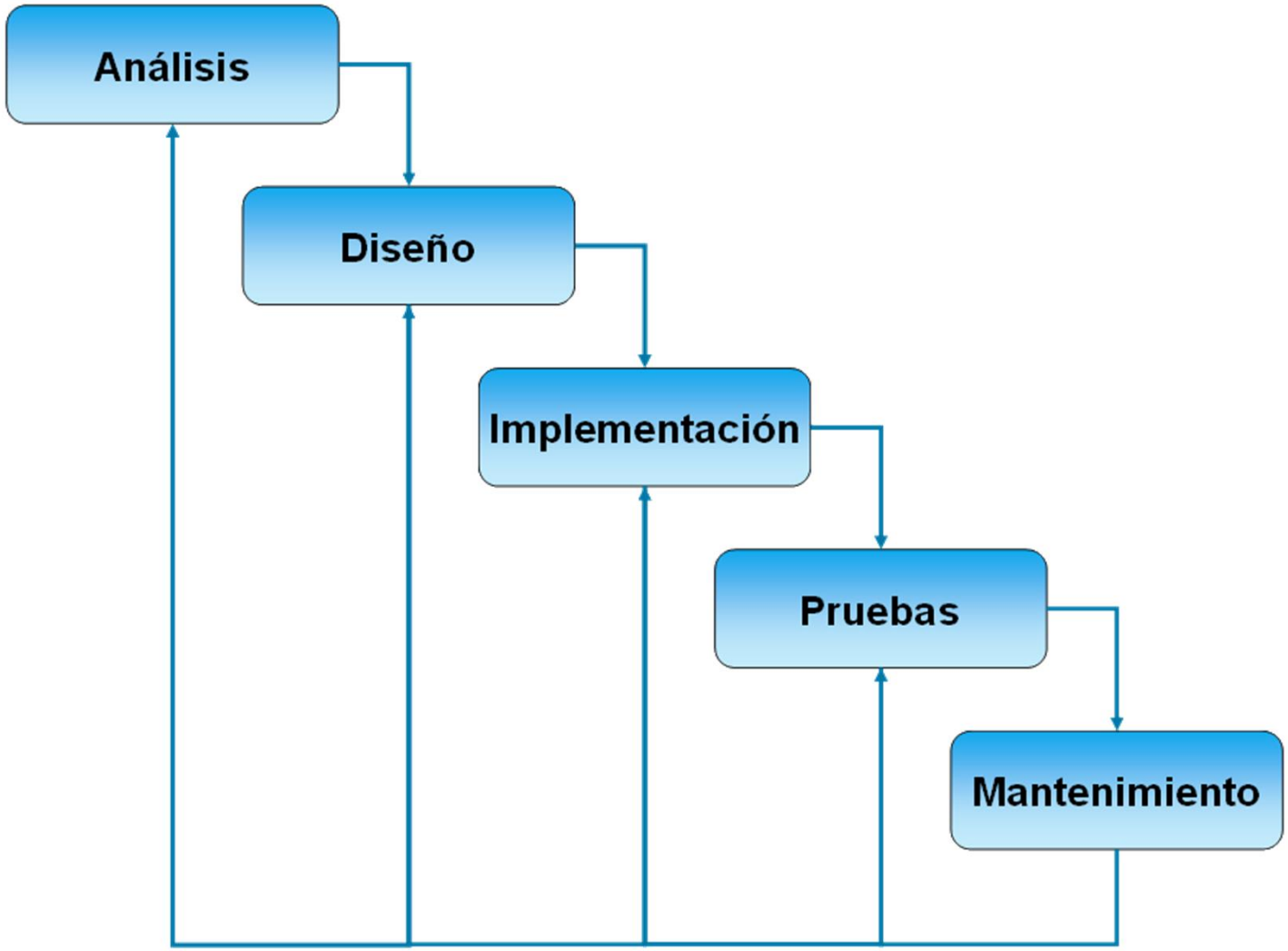
El modelo de ciclo de vida ayuda al desarrollador a escoger una estrategia para desarrollar el software. El paradigma de desarrollo software tiene su propio set de herramientas, métodos y procedimientos, los cuales son expresados de forma clara, y define el ciclo de vida del desarrollo del software.

Los modelos más importantes son: Cascada, Incremental y Evolutivo.

Ciclo de vida en Cascada

El modelo de cascada es el modelo de paradigma más simple en desarrollo de software. Sigue un modelo en que las fases del Ciclo de Vida funcionarán una detrás de la otra de forma lineal. Lo que significa que solamente cuando la primera fase se termina se puede empezar con la segunda, y así progresivamente.

Ciclo de vida en Cascada



Ciclo de vida en Cascada

Este modelo asume que todo se lleva a cabo y tiene lugar tal y como se había planeado en la fase anterior, y no es necesario pensar en asuntos pasados que podrían surgir en la siguiente fase. Este modelo no funcionará correctamente si se dejan asuntos de lado en la fase previa. La naturaleza secuencial del modelo no permite volver atrás y deshacer o volver a hacer acciones.

Este modelo es recomendable cuando el desarrollador ya ha diseñado y desarrollado softwares similares con anterioridad, y por eso está al tanto de todos sus dominios.

Ciclo de vida en Cascada

Los inconvenientes de este modelo son:

- Los proyectos reales raramente siguen el flujo secuencial que propone el modelo, siempre hay iteraciones y se crean problemas en la aplicación del paradigma.

- Normalmente, es difícil para el cliente establecer explícitamente al principio todos los requisitos. El ciclo de vida clásico lo requiere y tiene dificultades en acomodar posibles incertidumbres que pueden existir al comienzo de muchos productos.

- El cliente debe tener paciencia. Hasta llegar a las etapas finales del proyecto, no estará disponible una versión operativa del programa. Un error importante no detectado hasta que el programa este funcionando puede ser desastroso.

La ventaja de este método radica en su sencillez ya que sigue los pasos intuitivos necesarios a la hora de desarrollar el software.

Ciclo de vida Evolutivos

Los evolutivos son modelos iterativos, permiten desarrollar versiones cada vez más completas y complejas, hasta llegar al objetivo final deseado; incluso evolucionar más allá, durante la fase de operación. Los modelos “Iterativo Incremental” y “Espiral” (entre otros) son dos de los más conocidos y utilizados del tipo evolutivo.

La idea detrás de este modelo es el desarrollo de una implantación del sistema inicial, exponerla a los comentarios del usuario, refinarla en N versiones hasta que se desarrolle el sistema adecuado.

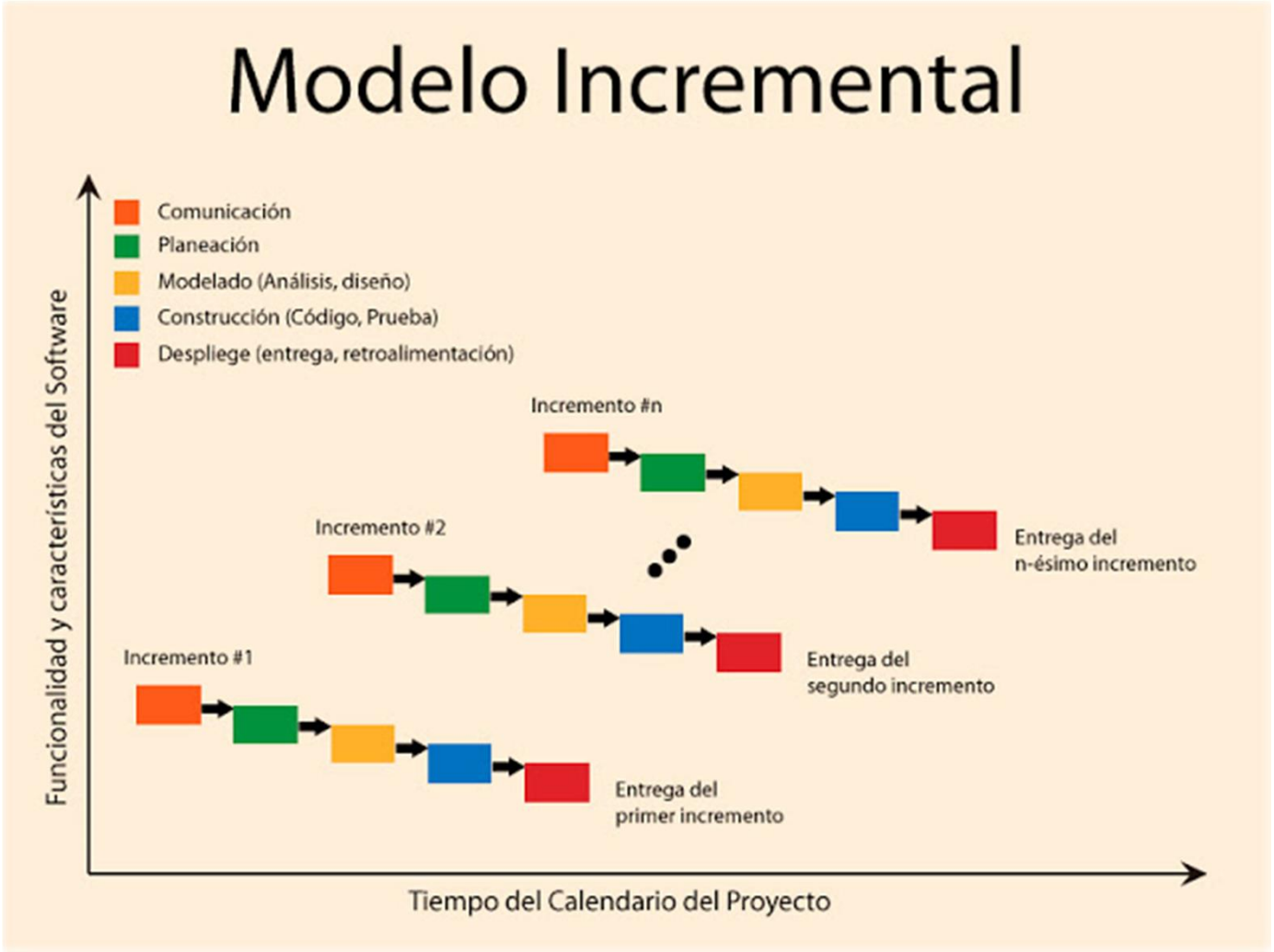
Ciclo de vida Evolutivos – Modelo Iterativo Incremental

El modelo incremental combina elementos del modelo en cascada con la filosofía interactiva de construcción de prototipos.

Se basa en la filosofía de construir incrementando las funcionalidades del programa. Este modelo aplica secuencias en cascada. Cada secuencia lineal produce un incremento del software.

Cuando se utiliza un modelo incremental, el primer incremento es a menudo un producto esencial, sólo con los requisitos básicos. Este modelo se centra en la entrega de un producto operativo con cada incremento. Los primeros incrementos son versiones incompletas del producto final, pero proporcionan al usuario la funcionalidad que precisa y también una plataforma para la evaluación.

Ciclo de vida Evolutivos – Modelo Iterativo Incremental



Ciclo de vida Evolutivos – Modelo Iterativo Incremental

Ventajas

Entre las ventajas que puede proporcionar un modelo de este tipo encontramos las siguientes:

- Mediante este modelo se genera software operativo de forma rápida y en etapas tempranas del ciclo de vida del software.
- Es un modelo más flexible, por lo que se reduce el coste en el cambio de alcance y requisitos.
- Es más fácil probar y depurar en una iteración más pequeña.
- Es más fácil gestionar riesgos.
- Cada iteración es un hito gestionado fácilmente

Ciclo de vida Evolutivos – Modelo Iterativo Incremental

Inconvenientes

Para el uso de este modelo se requiere una experiencia importante para definir los incrementos y distribuir en ellos las tareas de forma proporcionada. Entre los inconvenientes que aparecen en el uso de este modelo podemos destacar los siguientes:

- Cada fase de una iteración es rígida y no se superponen con otras.
- Pueden surgir problemas referidos a la arquitectura del sistema porque no todos los requisitos se han reunido, ya que se supone que todos ellos se han definido al inicio

Ciclo de vida Evolutivos – Modelo en Espiral

Las actividades de este modelo se conforman en una espiral, en la que cada bucle o iteración representa un conjunto de actividades. Las actividades no están fijadas a ninguna prioridad, sino que las siguientes se eligen en función del análisis de riesgo, comenzando por el bucle interior.

Para cada ciclo habrá cuatro actividades:

Determinar o fijar los objetivos. Se definen los objetivos específicos para posteriormente identifica las limitaciones del proceso y del sistema de software, además se diseña una planificación detallada de gestión y se identifican los riesgos.

Ciclo de vida Evolutivos – Modelo en Espiral

Análisis del riesgo. En este paso se efectúa un análisis detallado para cada uno de los riesgos identificados del proyecto, se definen los pasos a seguir para reducir los riesgos y luego del análisis de estos riesgos se planean estrategias alternativas.

Desarrollar, verificar y validar. En este tercer paso, después del análisis de riesgo, se eligen un paradigma para el desarrollo del sistema de software y se lo desarrolla.

Planificar. En este último paso es donde el proyecto se revisa y se toma la decisión si se debe continuar con un ciclo posterior al de la espiral. Si se decide continuar, se desarrollan los planes para la siguiente fase del proyecto.

Ciclo de vida Evolutivos – Modelo en Espiral

Ventajas

- El análisis del riesgo se hace de forma explícita y clara. Une los mejores elementos de los restantes modelos.
- Reduce riesgos del proyecto
- Incorpora objetivos de calidad
- Integra el desarrollo con el mantenimiento, etc.
- Además es posible tener en cuenta mejoras y nuevos requerimientos sin romper con la metodología, ya que este ciclo de vida no es rígido ni estático.

Inconvenientes

- Genera mucho tiempo en el desarrollo del sistema
- Modelo costoso
- Requiere experiencia en la identificación de riesgos

Fases del desarrollo de una aplicación

Análisis:
Esta es la primera fase del proyecto. Una vez finalizada, pasamos a la siguiente (diseño).
Es la fase de mayor importancia en el desarrollo del proyecto y todo lo demás dependerá de lo bien detallada que esté. También es la más complicada, ya que no está automatizada y depende en gran medida del analista que la realice.

Las tareas a realizar son:

- Entrevistas
- Planificación conjunta de aplicaciones
- Planificación conjunta de requisitos
- Brainstorming
- Prototipos
- Casos de uso

Fases del desarrollo de una aplicación

Se especifican y analizan los requisitos funcionales y no funcionales del sistema.

Funcionales: Qué funciones tendrá que realizar la aplicación. Qué respuesta dará la aplicación ante todas las entradas. Cómo se comportará la aplicación en situaciones inesperadas.

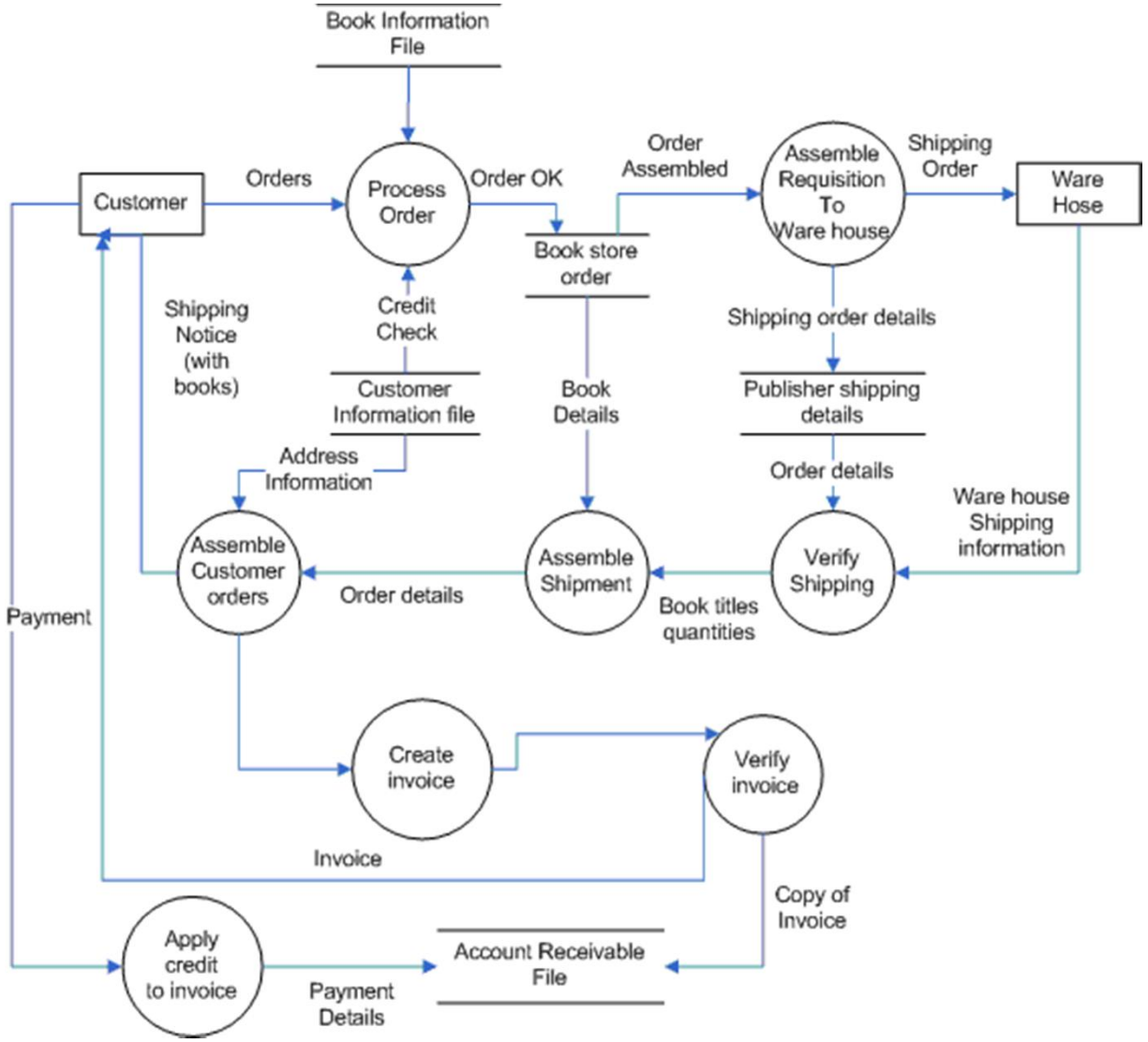
No funcionales: Tiempos de respuesta del programa, legislación aplicable, tratamiento ante la simultaneidad de peticiones, etc.

Fases del desarrollo de una aplicación

Para representar los requisitos se utiliza las diferentes técnicas:

Un diagrama de flujo de datos o DFD: (Es un diagrama que representa el flujo de datos entre los distintos procesos, entidades externas y almacenes que forman el sistema.
Los procesos identifican las funciones que el sistema debe cumplir.
Las entidades externas representan componentes que no forman parte del sistema (como usuarios o departamentos) pero proporcionan datos o los reciben.
Los almacenes representan los datos. Se define las estructuras de datos se debe almacenar y están representados con flechas que indican la dirección de entrada y salida de los mismos.

Fases del desarrollo de una aplicación



Fases del desarrollo de una aplicación

Para representar los requisitos se utiliza las diferentes técnicas:

Un diagrama de flujo de control o DFC: Similar al DFD con la diferencia de que muestra el flujo de control en lugar de datos

Un diagrama de flujo de estados DFE: Representa como se comporta el sistema como consecuencia de sucesos externos**.**

Diagrama Entidad / Relación: Representa los datos y la forma en que se relacionan entre ellos (1 a 1, 1 a muchos, muchos a muchos). Es MUY MUY importante que esté bien detallado.

Diccionario de datos: Es un conjunto de metadatos que contiene las características lógicas y puntuales de los datos y almacenes que se van a utilizar en el sistema presentes en el DFD.

Fases del desarrollo de una aplicación

Diseño:
Durante esta fase, donde ya sabemos lo que hay que hacer, el siguiente paso es ¿Cómo hacerlo?
Se debe dividir el sistema en partes y establecer qué relaciones habrá entre ellas además de decidir qué hará exactamente cada parte.

En definitiva, debemos crear un modelo funcional-estructural de los requerimientos del sistema global, para poder dividirlo y afrontar las partes por separado.

En este punto, se deben tomar decisiones importantes, tales como:

- Entidades y relaciones de las bases de datos. e va a utilizar.
- Selección del lenguaje de programación que s
- Selección del Sistema Gestor de Base de Datos.
- Etc.

Fases del desarrollo de una aplicación

Principalmente hay 2 tipos de diseño:

Diseño estructurado: Persigue elaborar algoritmos que cumplan la propiedad de modularidad. Para ello, dado un problema se busca dividir dicho programa en módulos siguiendo los **principios de diseño** de **descomposición por refinamientos sucesivos**, creación de una **jerarquía modular** y elaboración de módulos **independientes**.

Diseño Orientado a objetos: Un programa orientado a objetos se caracteriza por la interacción de esos objetos. El diseño orientado a objetos es la disciplina que define los objetos y sus interacciones para resolver un problema de negocio que fue identificado y documentado durante el análisis orientado a objetos.

Fases del desarrollo de una aplicación

Codificación:
Durante la fase de codificación se realiza el proceso de programación.

Consiste en elegir un determinado lenguaje de programación, codificar toda la información anterior y llevarlo a código fuente.

Esta tarea la realiza el programador y tiene que cumplir exhaustivamente con todos los datos impuestos en el análisis y en el diseño de la aplicación.

Las características deseables de todo código son:
- Modularidad: que esté dividido en trozos más pequeños.
- Corrección: que haga lo que se le pide realmente.
- Fácil de leer: para facilitar su desarrollo y mantenimiento futuro.
- Eficiencia: que haga un buen uso de los recursos.
- Portabilidad: que se pueda implementar en cualquier equipo.

Fases del desarrollo de una aplicación

Durante esta fase, el código pasa por diferentes estados:

Código Fuente: Es el escrito por los programadores en algún editor de texto. Se escribe usando algún lenguaje de programación de alto nivel y contiene el conjunto de instrucciones necesarias.

Código Objeto: Es el código binario resultado de compilar el código fuente. La compilación es la traducción de una sola vez del programa, y se realiza utilizando un compilador.
La interpretación es la traducción y ejecución simultánea del programa línea a línea.
El código objeto no es directamente inteligible por el ser humano, pero tampoco por la computadora. Es un código intermedio entre el código fuente y el ejecutable y sólo existe si el programa se compila, ya que si se interpreta (traducción línea a línea del código) se traduce y se ejecuta en un solo paso.

Fases del desarrollo de una aplicación

- **Código Ejecutable**: Es el código binario resultante de enlazar los archivos de código objeto con ciertas rutinas y bibliotecas necesarias. El sistema operativo será el encargado de cargar el código ejecutable en memoria RAM y proceder a ejecutarlo. También es conocido como código máquina y ya sí es directamente inteligible por la computadora.

Los programas interpretados no producen código objeto. El paso de fuente a ejecutable es directo.

Fases del desarrollo de una aplicación

Pruebas:
Un conjunto de actividades las cuyo objetivo es proporcionar información objetiva e independiente sobre la calidad del producto a la parte interesada. Es una actividad más en el proceso de control de calidad.
Dependiendo del tipo de pruebas, estas actividades podrán ser implementadas en cualquier momento de dicho proceso de desarrollo.
Ya que existen distintos modelos de desarrollo de software, así los hay de modelos de pruebas.

Fases del desarrollo de una aplicación

Pruebas:
Un conjunto de actividades las cuyo objetivo es proporcionar información objetiva e independiente sobre la calidad del producto a la parte interesada. Es una actividad más en el proceso de control de calidad.
Dependiendo del tipo de pruebas, estas actividades podrán ser implementadas en cualquier momento de dicho proceso de desarrollo.
Ya que existen distintos modelos de desarrollo de software, así los hay de modelos de pruebas.

Fases del desarrollo de una aplicación

Tipos de pruebas:

Pruebas estáticas: Son el tipo de pruebas que se realizan sin ejecutar el código de la aplicación. Se refiere en gran medida a la revisión de documentos, ya que no se hace una ejecución de código.

Fases del desarrollo de una aplicación

Tipos de pruebas:

Pruebas dinámicas: Todas aquellas pruebas que para su ejecución requieren la ejecución de la aplicación.
Las pruebas dinámicas permiten el uso de técnicas de caja negra y caja blanca con mayor amplitud. Debido a la naturaleza dinámica de la ejecución de pruebas es posible medir con mayor precisión el comportamiento de la aplicación desarrollada.

Todas las pruebas pueden ser manuales o automáticas (se utiliza otro software que analiza todas las variables posibles

Fases del desarrollo de una aplicación

Tipos de pruebas:

- Pruebas de Compatibilidad: Comprobarán que tu desarrollo es compatible con todos los navegadores y todos los sistemas convenientes. Estas pruebas son realmente importantes para que tu producto llegue a todos los usuarios que deberían de llegar y que todo el mundo pueda utilizarlo con lo que disponga en su equipo informático.

Fases del desarrollo de una aplicación

Mantenimiento:
Es la modificación de un producto de software después de la entrega, para corregir errores, mejorar el rendimiento, u otros atributos.

El mantenimiento del software es una de las actividades más comunes en la ingeniería de software y se puede resumir como la evolución del software.

Lenguajes de Programación

Los lenguajes de programación son los que nos permiten comunicarnos con el hardware del ordenador.

En otras palabras, es muy importante tener muy clara la función de los lenguajes de programación. Son los instrumentos que tenemos para que el ordenador realice las tareas que necesitamos.
Hay multitud de lenguajes de programación, cada uno con unos símbolos y unas estructuras diferentes. Además, cada lenguaje está enfocado a la programación de tareas o áreas determinadas. Por ello, la elección del lenguaje a utilizar en un proyecto es una cuestión de extrema importancia.
Los lenguajes de programación han sufrido su propia evolución, como se puede apreciar en la figura siguiente:

Lenguajes de Programación

Características de los Lenguajes de Programación (según su nivel de abstracción)

- **Lenguaje máquina**:

  - Sus instrucciones son combinaciones de unos y ceros.
  - Es el único lenguaje que entiende directamente el ordenador. (No necesita traducción).
  - Fue el primer lenguaje utilizado.
  - Es único para cada procesador (no es portable de un equipo a otro).
  - Hoy día nadie programa en este lenguaje.

Lenguajes de Programación

Características de los Lenguajes de Programación (según su nivel de abstracción)

- **Lenguaje ensamblador**:

    - Sustituyó al lenguaje máquina para facilitar la labor de programación.
    - En lugar de unos y ceros se programa usando mnemotécnicos (instrucciones complejas).
    - Necesita traducción al lenguaje máquina para poder ejecutarse.
    - Sus instrucciones son sentencias que hacen referencia a la ubicación física de los archivos en el equipo.
    - Es difícil de utilizar.

Lenguajes de Programación

Características de los Lenguajes de Programación (según su nivel de abstracción)

- **Lenguaje de alto nivel basados en código**:
  - Sustituyeron al lenguaje ensamblador para facilitar más la labor de programación.
  - En lugar de mnemotécnicos, se utilizan sentencias y órdenes derivadas del idioma inglés. (Necesita traducción al lenguaje máquina).
  - Son más cercanos al razonamiento humano.
  - Son utilizados hoy día, aunque la tendencia es que cada vez menos.

Lenguajes de Programación

Características de los Lenguajes de Programación (según su nivel de abstracción)

- **Lenguajes visuales**:

  - Están sustituyendo a los lenguajes de alto nivel basados en código.
  - En lugar de sentencias escritas, se programa gráficamente usando el ratón y diseñando directamente la apariencia del software.
  - Su correspondiente código se genera automáticamente.
  - Necesitan traducción al lenguaje máquina.
  - Son completamente portables de un equipo a otro.

Lenguajes de Programación

Características de los Lenguajes de Programación (según su forma de ejecución)

• **Lenguajes compilados**: Son aquellos en los que el programa es traducido a código máquina por completo antes de su ejecución. La herramienta que produce esta transformación se denomina "compilador". Una vez generado el código máquina éste se puede ejecutar tantas veces como se desee. Si en el programa original hay errores, se detectan por el compilador y no se genera el ejecutable. Si un programa está dividido en diferentes archivos, éstos se compilan individualmente generando archivos "objeto" que después se enlazarán o "linkaran" para generar el fichero ejecutable final. El proceso de compilación es relativamente lento. Un programa compilado para una plataforma o sistema operativo no funcionará en otro sistema operativo a menos que se vuelva a compilar.
Ejemplo: C, Pascal, Java, Delphi, Fortran, Ada, Algol, ect…

Lenguajes de Programación

Características de los Lenguajes de Programación (según su forma de ejecución)

• **Lenguajes interpretados**: Son aquellos en los que el programa fuente es ejecutado conforme va siendo traducido por bloques y no por completo. Esto supone que la ejecución del programa es más lenta que en el caso anterior. Sin embargo, los programas interpretados son fácilmente portables entre diferentes plataformas.

Ejemplo: SQL, HTML, XML, PHP, ASP, Java Script, LISP, Perl, Python, ect..

Lenguajes de Programación

Características de los Lenguajes de Programación (según su paradigma)

• **Imperativos**: Son lenguajes que permiten escribir programas basados en algún algoritmo de resolución de un problema. Por lo tanto, deben especificar cada uno de los pasos que nos lleva a la solución. También se les conoce como lenguajes de programación procedimentales ya que se ha de establecer el procedimiento para alcanzar la solución deseada. Todos los lenguajes anteriormente descritos desde la primera hasta la tercera generación son lenguajes de programación imperativos (incluidos los orientados a objetos).

Lenguajes de Programación

Características de los Lenguajes de Programación (según su paradigma)

• **Declarativos**: En el programa se indica qué es lo que se desea obtener y no cómo hacerlo. En los lenguajes declarativos las sentencias que se utilizan lo que hacen es describir el problema que se quiere solucionar, pero no las instrucciones necesarias para solucionarlo.
Sin embargo, la programación declarativa se ha dividido tradicionalmente en otros dos subconjuntos:

La programación funcional basada en la utilización de funciones matemáticas. El lenguaje más representativo de este paradigma es LISP.

La programación lógica basada en la lógica de cálculo de predicados de primer orden. Se especifican las condiciones que satisfacen las soluciones. El lenguaje más representativo de este tipo es PROLOG.

Lenguajes de Programación

Características de los Lenguajes de Programación (según su ámbito de aplicación)

• **Propósito general**: Se aplican a la solución de cualquier tipo de problema. Pueden nombrarse lenguajes como C, C++, Pascal, BASIC, Java, Delphi, Modula, etc.

• **Bases de Datos**: Ya sean imperativos o declarativos, 4GL o de tercera generación, su finalidad es la manipulación de bases de datos. Los más conocidos son SQL, QBE, QUEL, FoxPro, Clipper, dBase, Cobol, etc.

• **Desarrollo Web**: Algunos lenguajes de desarrollo web son: HTML, XML, Visual Basic Script, Java, JavaScript, ASP, PHP, Perl, etc.

• **Otras finalidades**: De uso más específico y, por lo tanto, menos extendido podemos nombrar FORTRAN en el ámbito científico, LISP o PROLOG aplicados a la inteligencia artificial.