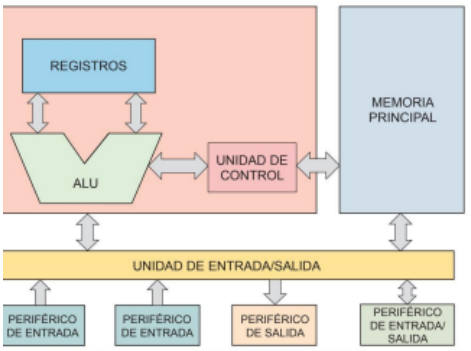
RESUMEN T1 ENTORNOS DE DESARROLLO

1.1 El software y su relación con otras partes del ordenador.

SW es el conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora. Partes del ordenador relacionadas:



1. Unidad central de procesos: (CPU): se compone de la Unidad Aritmético-lógica (ALU), Unidad de Control (UC) y los Registros.
2. Memoria principal o RAM: Contiene las instrucciones del programa a ejecutar.
3. Unidad de entrada/salida: Permite la comunicación con los periféricos

Según la función del software hay 3 tipos:

1. Software de sistemas.
2. Software de programación o desarrollo.
3. Software de aplicación.

1.2 Lenguajes de programación.

Los elementos básicos de los lenguajes de programación son:

- Identificadores
- constantes
- operadores
- instrucciones
- comentarios.

Existen 3 tipos de lenguajes:

- Lenguajes bajo nivel o lenguaje máquina (binario)
- Lenguajes intermedios o ensambladores (Intérpretes y compiladores)
- Lenguajes de alto nivel (programación).

1.3 Código fuente, objeto y ejecutable.

1. Código Objeto. Es el que entiende el ordenador (binario)
2. Código Fuente. Es el que escribe el programador en lenguaje de alto nivel. Para poder traducirlo a código objeto, existen:
    - *Compiladores*: traducen todo el código fuente de una vez, si existe error no se genera código objeto. Una vez terminado se crea un código ejecutable.
    - *Intérpretes*. hacen a la vez traducción y ejecución de partes del código fuente. Más fácil de detectar errores.

3. Código Ejecutable: comprende un conjunto de instrucciones compiladas y enlazadas, listas para ser ejecutadas por una computadora. Si no se ha generado este tipo de código en la traducción será necesario un enlazador.

Hoy en día existen los entornos de desarrollo integrados en la aplicación de programación que tienen todas las herramientas para generar los 3 tipos de códigos y reproducir el ejecutable.

1.4 Máquinas Virtuales.

Existen 2 tipos:

- Máquinas virtuales de sistema: emulan un ordenador por completo pudiendo instalarse un SO en su interior. Se puede trabajar en ella como si fuera una máquina real.
- Máquinas virtuales de proceso: ejecuta un proceso concreto dentro de un SO. El objetivo es ejecutar este proceso en cualquier plataforma. El proceso no es ejecutado por la CPU del ordenador si no por la máquina virtual. Por ejemplo JAVA

    En este último tipo de máquinas virtuales el proceso para poder ejecutar el programa es:

1. El programador escribe Código Fuente del programa en el editor de textos Java y se genera archivo de texto .java
2. Se compila el programa fuente, por el compilador javac, genera extensión .class
3. La Máquina virtual de Java traduce el archivo compilado a código binario para poder ser ejecutado.

1.5 La ingeniería del software.

En los 70 crisis del software. Hay distintos problemas fundamentales:

- Planificación y estimación de costes.
- Baja productividad.
- Baja calidad del software.

Fitz Bauer propuso establecer y usar unos principios de ingeniería para llegar a obtener software rentable, fiable y que funcione eficientemente en máquinas reales.

1.6 Fases del desarrollo de software.

- Análisis: analiza necesidades del usuario. Determina que debe hacer el programa: qué información será procesada, qué función y rendimiento se desea. Existen requisitos no funcionales: *Fiabilidad* (no fallos). *Escalabilidad* (manejar aumentos de carga sin perder rendimiento). *Extensibilidad* (capacidad para añadir nuevas funciones). *Seguridad* (protege la información). *Mantenibilidad* (grado en el que el SW es comprendido, reparado y mejorado).
- Diseño: Se establece cómo se va a resolver el problema planteado por el análisis, dando funciones al hardware. Refinamiento de la fase anterior.
- Programación: Consiste en traducir el diseño de forma legible para el ordenador. Se escribe código fuente para cada función de cada elemento de software implicado. El resultado es un código ejecutable. Elementos importantes: Declaración de variables. Comentarios. Nombres de clases, atributos, etc. Líneas en blanco. Sangrados.
- Pruebas: Se comprueba si el programa funciona correctamente. Existen Pruebas unitarias (por separado cada función), pruebas de integración (se van añadiendo poco a poco componentes) y pruebas de validación (se prueba el programa completo).

  Técnicas de pruebas: Caja blanca o estructurales: se examina cada módulo del código fuente bucles,etc. Caja negra o funcionales: se examina las entradas y las salidas del programa
- Explotación: Se instala y se pone en marcha en el entorno del usuario.
- Mantenimiento: Puede sufrir cambio por distintos motivos. *Correctivo* (corrige errores que detecta el cliente). *Adaptativo* (Se debe adaptar a cambio en SO,etc). *Perfectivo* (el cliente solicita funciones adicionales o requisitos no funcionales como la velocidad o seguridad).

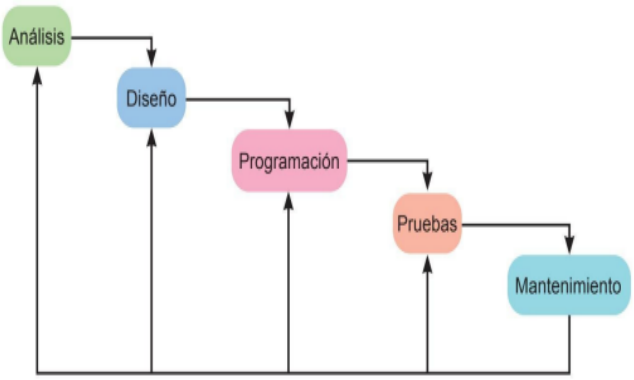
1.7 Roles en el proceso de desarrollo del software.

- Jefe de proyecto (encargado de la gestión, máximo responsable)
- Expertos del dominio (empleado del cliente que conoce el problema)
- Analista o analista funcional (crea modelo claro y consistente).
- Arquitecto (define las líneas maestras y establece arquitectura del sistema)
- Diseñador (diseña partes del sistema que implementan requisitos con detalle)
- Programador (escribir código fuente a partir del diseño)
- Probador (realiza pruebas para garantizar la calidad)
- Encargado de la implantación (encargado de instalación y puesta en marcha de los programas en el entorno del cliente)

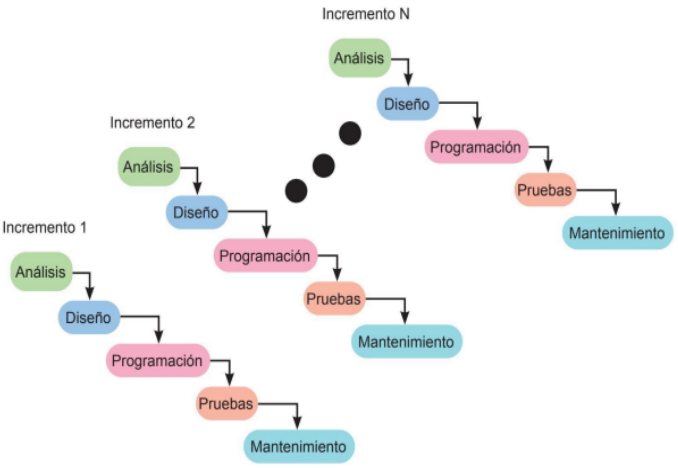
1.8 Ciclos de vida del Software.

Determina el orden que se deben de llevar a cabo las tareas del desarrollo del software.

- Modelo en cascada: También llamado modelo de vida clásico. Enfoque sistemático y secuencial. Para pasar de una fase a otra se deben cumplir ciertos objetivos. Modelo sencillo de comprender. Presenta problemas como:
    1. Cambios generan confusión en el equipo del proyecto.
    2. Si en la fase de programación no se ha incorporado algún requisito o se ha realizado de forma incorrecta, será necesario rehacer todas las fases previas.
    3. el cliente debe esperar para tener una versión funcional hasta que el proyecto esté muy avanzado.

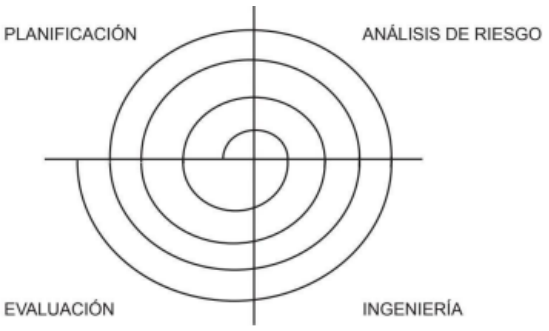


- Modelo de proceso incremental: El Software se divide en una serie de incrementos, en cada uno de los cuales se aplican los pasos del ciclo de vida clásico. Tiene como ventajas:
    1. Reduce los riesgos de retrasos, ya que se realizan diversas entregas.
    2. Los entregables intermedios aumentan la comunicación con los clientes que van detectando problemas o falta de requisitos.
    3. Permiten al usuario validar el sistema a medida que se construye.



- Modelos de proceso evolutivo: Estos modelos son iterativos y se caracterizan por el desarrollo de versiones cada vez más completas del SW.
    - ☐ Construcción de prototipos: Permite probar experimentalmente ciertas soluciones parciales a las necesidades del usuario. El objetivo principal es que los desarrolladores conozcan exactamente lo que tiene que hacer la aplicación. se construye a través de versiones cada vez más completas del SW.

    - ☐ Modelo en espiral: diseñado para cubrir las mejores características del modelo en cascada y construcción de prototipos. Se desarrolla en una serie de entregas, en las primeras puede ser un prototipo.



1.9 Metodologías de desarrollo de software.

Las metodologías han ido evolucionando con el paso del tiempo:

1. Desarrollo convencional: No seguían metodología, generaba muchos problemas. Desembocó en la llamada crisis del software.
2. Metodologías convencionales: primera respuesta a la crisis. Aparece la programación estructurada que abarca la totalidad del ciclo de vida del software.
3. Metodologías orientadas a objetos: surgen lenguajes orientados a objetos y métodos de diseño y análisis de objetos.

Las metodologías más usadas son:

Proceso Unificado de Rational.

Cada ciclo se compone de 4 fases:

- Fase de comienzo: objetivo estudiar la viabilidad del sistema, establecer el objetivo del sistema y delimitar su alcance. Al finalizar esta fase se decide si el proyecto continúa o no.
- Fase de elaboración: Objetivo analizar el dominio del problema y establecer la estructura del software, eliminar riesgos y establecer el plan del proyecto.
- Fase de construcción: se desarrolla el proyecto de forma incremental hasta que esté listo. En cada iteración se seleccionan ciertos casos de uso y se refinan.
- Fase de transición: Pone en funcionamiento el sistema y se pone a disposición de los usuarios. Suele iniciarse con una fase beta.

Al término de cada una de las fases, la dirección decidirá si el proyecto puede continuar con la siguiente fase. Se deben cumplir hitos o plazos de tiempo en cada fase.

Modelos de desarrollo Ágil.

se rige por 4 puntos fundamentales:

1. Los individuos y sus interacciones **VS** los procesos y las herramientas.
2. El software que funciona **VS** la documentación exhaustiva.
3. Colaboración con el cliente **VS** negociación del contrato.
4. Responder ante el cambio **VS** el apego a un plan.

Existen modelos concretos de metodología ágil:

- ☐ Programación Extrema: se rige por 5 valores: Simplicidad:simplificar el diseño para agilizar el desarrollo y facilitar el mantenimiento. Comunicación: comunicación cercana e informal con el cliente para mayor retroalimentación. Retroalimentación: al realizar entregas al cliente, en cortos intervalos de tiempo se obtiene retroalimentación que evita rehacer partes del trabajo. Valentía: no se debe diseñar pensando en requisitos futuros. Respeto: respetarse entre miembros del equipo de desarrollo.
- ☐ Scrum: se basa en tres pilares: Transparencia: Todos los miembros del equipo conocen en todo momento qué ocurre y cómo. Esto da visión global del proyecto. Inspección: todos los miembros inspeccionan el proyecto de manera autoorganizada. Adaptación: el equipo debe adaptarse a los posibles cambios. Esta metodología se divide en sprints (periodo de tiempo para cada actividad, máximo 1 mes aconsejable 2 semanas). Diariamente se realizan reuniones de 15 minutos para saber cómo continuar diariamente.