**¿Qué es el Software?**

Trata de crear productos y aplicaciones con ciertos objetivos específicos. Utilizando muchas técnicas y herramientas que exigen mucha capacidad intelectual, lo cual hace de la creación del software un proceso complejo.

Se puede decir que es conocimiento acumulado, empaquetado y ejecutable.

**Se define a partir de los propios aspectos que toma:**

* Ingeniería de requisitos.
* Análisis y Diseño.
* Programa.
* Pruebas.
* Manuales – Capacitación.

Un software puede cuenta con uno a varios programas, procedimientos, una documentación, y los datos de la operación del sistema.

**¿Por qué es único Software?**

* Es intangible.
* Posee una gran cantidad de intelecto.
* Es modificable hasta el infinito.
* Su proceso de desarrollo es intensivo, y se basa en equipos y proyectos.

**Características del software.**

* **Corrección funcional:** Esta de acuerdo con las especificaciones que fueron definidas.
* **Confiabilidad:** El usuario puede confiar y depender del software.
* **Robustez:** Esta preparado para situaciones imprevistas.
* **Performance**: El software es eficiente.
* **Amistosidad:** Es versátil y accesible.
* **Verificabilidad:** Se puede confirmar fácilmente la viabilidad del software.
* **Mantenibilidad:** Tiene la capacidad de codificase y mejorar con el tiempo.
* **Reusabilidad:** Permite crear otros programas sin reinventarlo por completo.
* **Portabilidad:** Puede funcionar en varios dispositivos.
* **Comprensibilidad:** Facilidad para entenderlo su funcionamiento.
* **Interoperatividad:** es compatible con otros sistemas.
* **Oportunidad:** Se puede realizar una aplicación en un tiempo estimable.

No siempre se puede optar por todas, es necesario hacer una balanza.

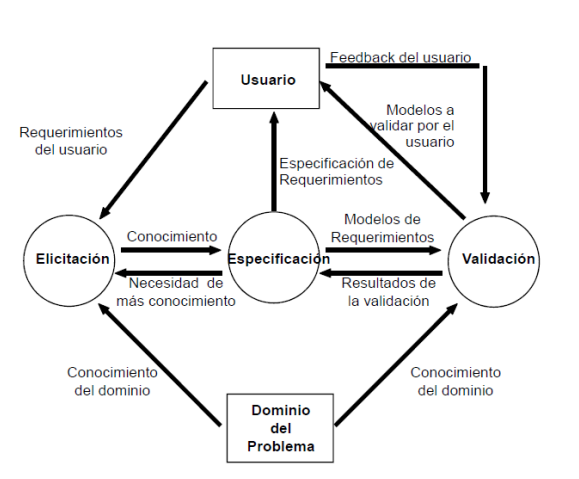
Para una buena documentación del código podría utilizarse diferentes herramientas, como los comentarios, variables auto explicativas, buena modularización o incluso un documento aparte que índique cómo funciona el código.

**Ingeniería de Software.**

Es una disciplina que utiliza diferentes métodos de la ingeniería para la producción, operación y mantenimiento del software. Utilizado para estimar tiempos y costos.

Podemos llegar a tener problemas cuando cometemos errores, mientras más arrastremos esos errores, más difícil y costoso se vuelve repáralo. No es lo mismo un error de una unidad de programación que se puede corregir con una actualización, que un error en los objetivos del software lo cual puede ser tan grave que tengamos que reiniciar todo el proceso.

**Ingeniería de Requisitos.**

****

Cuando hablamos de requisitos, hablamos de un ciclo donde el usuario nos da los requerimientos que desea y con ese conocimiento podemos realizar una especificación más detallada de lo que se quiere, creando modelos que deben ser validados tanto por nosotros como por el usuario.

El feedback del usuario puede ayudarnos a detectar fallas en la especificación o en los modelos como también la necesidad de obtener más detalles sobre ciertos aspectos.

También es importante destacar el dominio del problema, ya que este también nos provee cierto conocimiento a la hora de hacer especificaciones, no hay que quedarse solamente con lo que el usuario pide (algunas veces puede no saberlo), es necesario entender su contexto.

**El contrato social de los requerimientos – Derechos del usuario.**

* Debemos comunicar las ideas sin muchos tecnicismos.
* Hay que aprender sobre el negocio para el cual trabajamos.
* Escribir una especificación o contrato del software.
* Tratar de manera agradable a los clientes.
* Tener varias alternativas tanto para los requisitos como para su resolución.
* Estimar los mejor posible el costo de cualquier cambio en el software.
* Realizar un sistema que cumpla con los requisitos y tenga calidad.

**Verificación:** Que la aplicación funcione correctamente con los requerimientos especificados.

**Validación:** Que la aplicación cumpla con lo que el usuario espera.

Para realizar verificaciones y validaciones necesitamos hacer diferentes pruebas:

Escala de tiempo

Descripción generada automáticamente

El software compuesto de diferentes unidades.

1. Se comprueba el funcionamiento por los módulos unitarios, realizando una prueba de integración que verifique las especificaciones del diseño.
2. Con los módulos integrados podemos realizar una prueba funcional, para confirmar que el sistema cumple con los requerimientos funcionales.
3. Luego una prueba de desempeño para verificar eficiencia, o ciertas características con las cuales también debe de cumplir.
4. Si el software funciona solamente queda que el cliente acepte el sistema.
5. Si el sistema es aceptado se debe realizar la instalación para luego ponerlo a disposición de uso.

Diagrama

Descripción generada automáticamente con confianza baja

Se puede ver como un proceso de construcción y pruebas para cada etapa del desarrollo del software. Desde las especificaciones hasta el código y de ahí con diferentes pruebas de unidades hasta el sistema completo.

Escala de tiempo

Descripción generada automáticamente

Es más notorio esta rutina cuando un software debe perdurar en el tiempo, ya que este proceso se repite para cada versión del producto.

**¿Quién verifica?**

* Pruebas Unitarias: Las deben hacer los desarrolladores del mismo módulo ya que tienen más conocimiento de los detalles de su implementación.
* Pruebas de Integración: Las puede hacer el equipo de desarrollo. Requiere de conocimiento de interfaces y funciones.
* Resto de las pruebas: Lo suele realizar un equipo especializado que tenga una visión más global y que conozca los requerimientos del sistema.

**¿Por qué un equipo especializado?**

* Está familiarizado con herramientas para las pruebas.
* Tienen experiencia con el equipo de programación.
* El desarrollador del programa puede inconscientemente no buscar errores en su programa. Repetir sólo los errores que están corregidos. Y varios problemas de psicología de pruebas.

**Verificación Estática – Dinámica**

**Inspecciones del software:** verificación del sistema en forma estática. Se puede realizar **analizando el código fuente o realizando un análisis formal del sistema.**

**Pruebas del software:** verificación dinámica por medio de la observación del comportamiento del producto. Se realiza con datos de pruebas y utilizando técnicas de cajas.

**Caja negra:** Entrada a una caja donde no se conoce el contenido observando su salida.

Imagen que contiene Rectángulo

Descripción generada automáticamenteTiene varias características:

* No necesita el código fuente

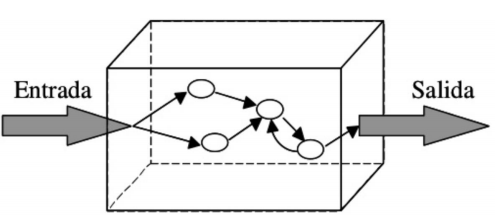
del programa.

* Se tiene en cuenta la

especificación.

* Partes del código ocultas

que no se utilizan.

**Caja blanca:** Conociendo las características del programa se pueden realizar mejores pruebas:

* Se necesita del código fuente.
* Tiene en cuenta los detalles

de la implementación.

* Puede que no se ejecute una

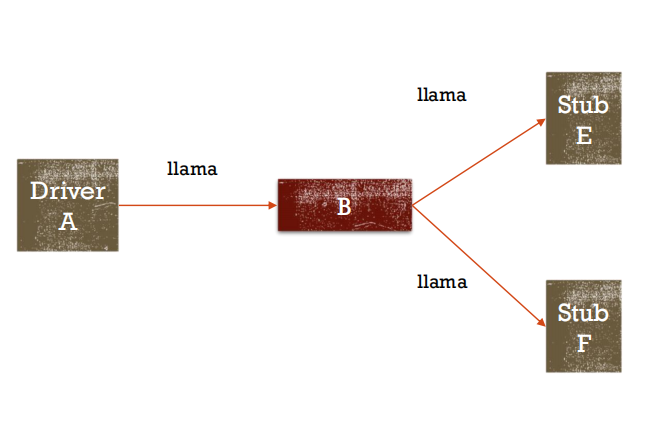
prueba que venga en las

especificaciones.

**Ejecución simbólica:** Se utilizan técnicas híbridas.

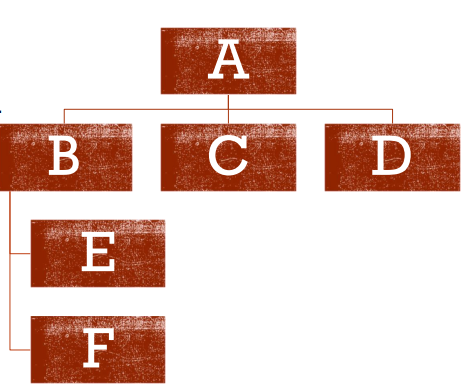
El **testing** no puede realizar todas las pruebas existentes, pero puede ayudar a mejorar el por mucho el sistema original. Casi siempre revela más la presencia de fallas que la ausencia y ayuda a encontrar su origen.

Para realizarse debe utilizar un subconjunto de las entradas (test set) y de ser posible el test debe ser repetible (difícil en el ámbito concurrente). Suele sugerirse realizar primero las pruebas de caja negra y cuando sean correctas utilizar las pruebas de caja blanca.

**Pruebas de Módulos.**

Cuando se realiza una prueba de un módulo supongamos el B, el cual es llamado por otro módulo A, se debe simular la llamada, cuando se hace eso, se llama a “A” Driver. Normalmente es este quien suministra los datos para la prueba. Cuando simulamos la llamada a los procedimientos “E” y “F” de los cuales depende B, se llaman a esos módulos Stub.

La prueba se realiza con métodos de prueba unitaria.

**Pruebas de Integración – Estrategias.**

* No incremental: Bing-bang.
* Incremental:

1. Bottom-Up.
2. Top-Down.
3. Sándwich.
4. Por disponibilidad.