



Ministerio de Educación y Deportes

Subsecretaría de Servicios Tecnológicos y
Productivos



Características del Software

1. El software complejo
2. El software es intangible y su intangibilidad es lo que contribuye fuertemente a su complejidad
3. El software es complejo, complejo de construir, complejo de entender, complejo de explicárselo a otros.
4. El software es desarrollado y no construido. La falta de principios y componentes bien definidos hace de él un producto diferente a cualquier otro
5. La existencia de pedidos de cambio y evolución constante de su función y/o estructura, su desarrollo propenso a errores, la dificultad para su estimación, la falta de comprensión de las implicancias que trae aparejado el cambio, son algunas de las razones que fundamenten su complejidad

Software

Conjunto de:

- *Programas*
- *Procedimientos*
- *Reglas*
- *Documentación*
- *Datos*



Software, en general, es un set de programas y la documentación que acompaña

- Existen tres tipos básicos de software.
Estos son:
 - System software
 - Utilitarios
 - Software de Aplicación





Organización del Software: Ingeniería de Software

Ingeniería de Software (IS): La aplicación de un enfoque sistemático, disciplinado y cuantificable para el desarrollo, operación y mantenimiento del software; es decir, la aplicación de la ingeniería al software.

El objetivo de la IS es convertir el desarrollo de software en un proceso formal:

- con resultados predecibles,
- que permitan obtener un producto final de alta calidad
- que satisfaga las necesidades y expectativas del cliente.



Ingeniería de Software - Tecnología Multicapa



La gestión de calidad fomenta una cultura continua de mejoras de procesos que conduce al desarrollo de enfoques cada vez más robustos.



Ingeniería de Software - Tecnología Multicapa



El **proceso** define un marco de trabajo para un conjunto de áreas claves, las cuales forman la base del control de gestión de proyectos de software



Ingeniería de Software - Tecnología Multicapa



Los **métodos** definen cómo construir técnicamente el software/ Abarcan una gran gama de tareas y dependen de un conjunto de principios básicos que gobiernan cada área de la tecnología e incluyen actividades de modelado y otras técnicas descriptivas.



Ingeniería de Software - Tecnología Multicapa

Herramientas (CASE)

Métodos

Las **herramientas** proporcionan un soporte automático o semi-automático para el proceso y los métodos, denominadas herramientas CASE (Computer-Aided Software Engineering)

un enfoque de calidad



Ingeniería de Software

El objetivo de la Ingeniería de Software es lograr productos de software de calidad (tanto en su forma final como durante su elaboración), mediante un proceso apoyado por métodos y herramientas



Disciplinas en la Ingeniería de Software



Disciplinas Técnicas

- Requerimientos
- Análisis y Diseño
- Implementación
- Prueba
- Despliegue

Construcción del
Producto



Disciplinas de Gestión

- Planificación de Proyecto
- Monitoreo y Control de Proyectos

Guía para la
construcción
del software



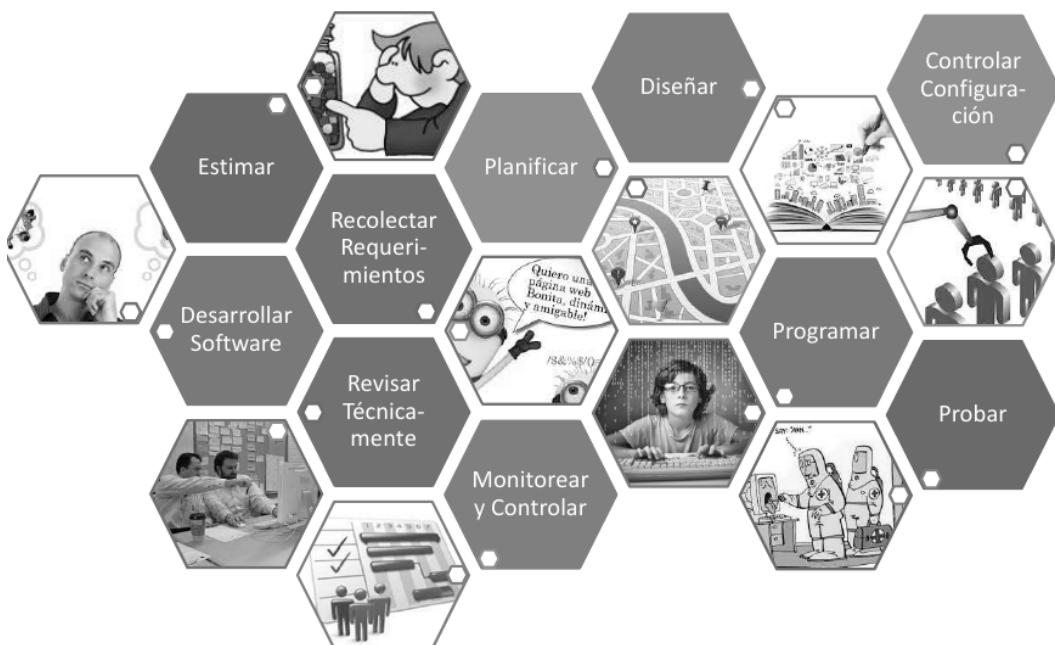
Disciplinas de Soporte

- Gestión de Configuración de Software
- Aseguramiento de Calidad
- Métricas

Brindan
herramientas
para la calidad

Desarrollo de Software

El desarrollo de software se trata de desarrollar y entregar gran software.
Un software es un gran software cuando complace a quienes lo necesitan, a quienes lo pidieron, a quienes lo usarán



Disciplinas Técnicas - Construcción del Software



Disciplinas Técnicas

- Requerimientos
- Análisis y Diseño
- Implementación
- Prueba
- Despliegue

Requerimientos: característica que el producto debe satisfacer



Disciplinas Técnicas - Construcción del Software



Disciplinas Técnicas

- Requerimientos
- Análisis y Diseño
- Implementación
- Prueba
- Despliegue

Análisis y diseño: diseñar es crear, tomar decisiones respecto de cuál es la mejor forma de satisfacer los requerimientos definidos para el producto.



Disciplinas Técnicas - Construcción del Software



Disciplinas Técnicas

- Requerimientos
- Análisis y Diseño
- Implementación
- Prueba
- Despliegue

Implementar: escribir código, que permitirá controlar lo que una computadora hará



Disciplinas Técnicas - Construcción del Software



Disciplinas Técnicas

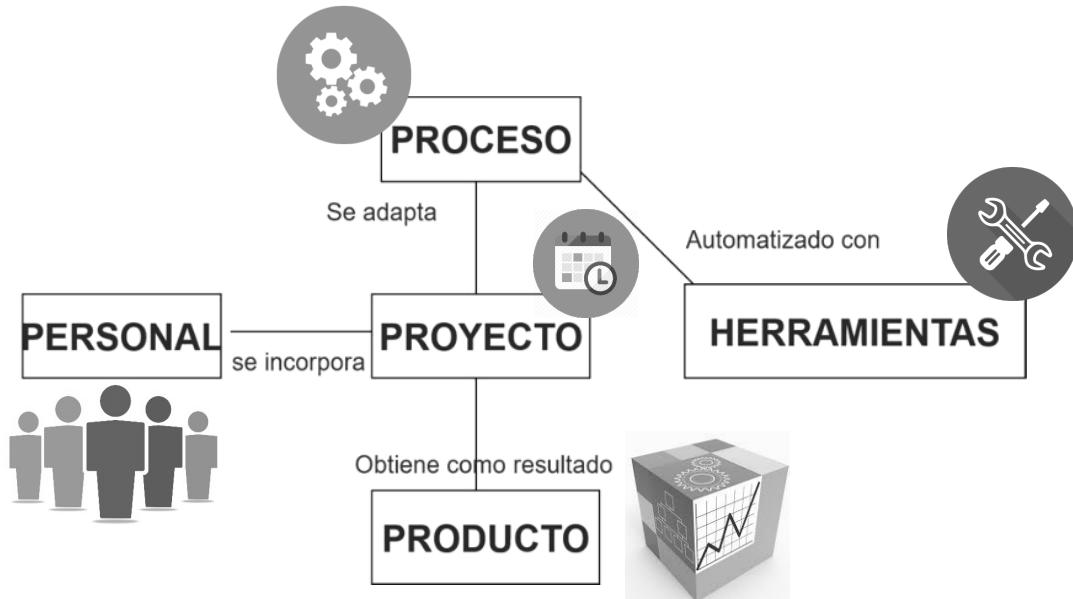
- Requerimientos
- Análisis y Diseño
- Implementación
- Prueba
- Despliegue

Prueba: Tiene como objetivo encontrar defectos. Se deberá validar que el producto que se está probando es el que el usuario quería y verificar que el producto funciona correctamente.

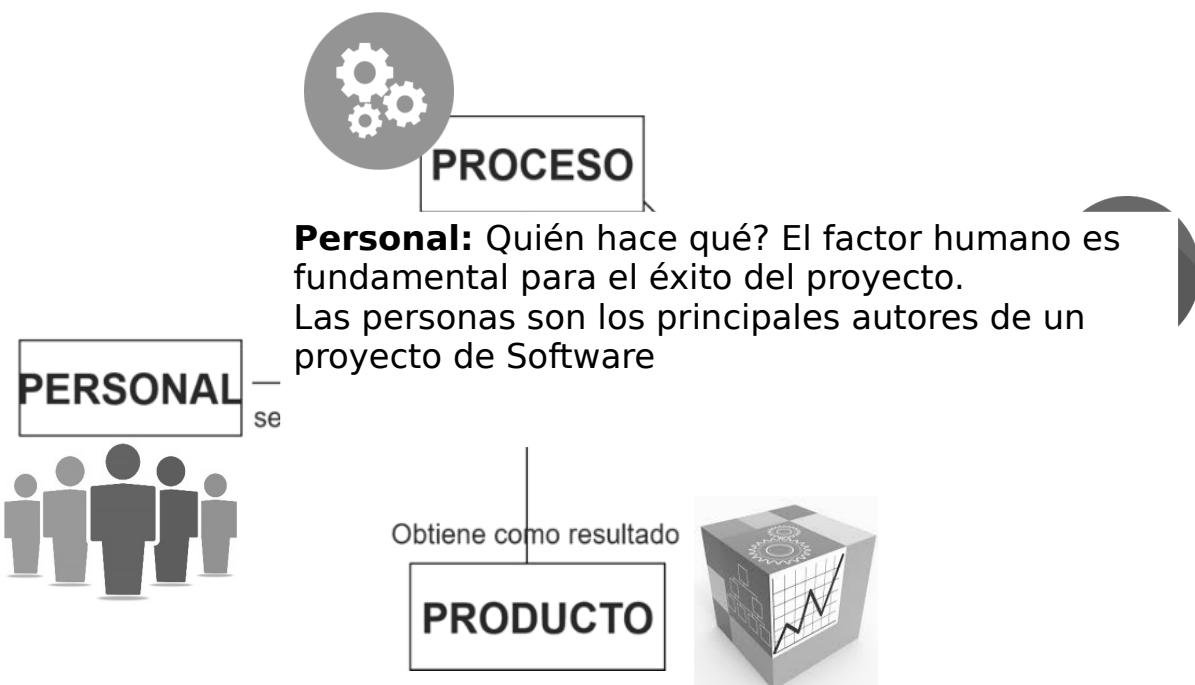


Desarrollo de Software - Las 4 “P”

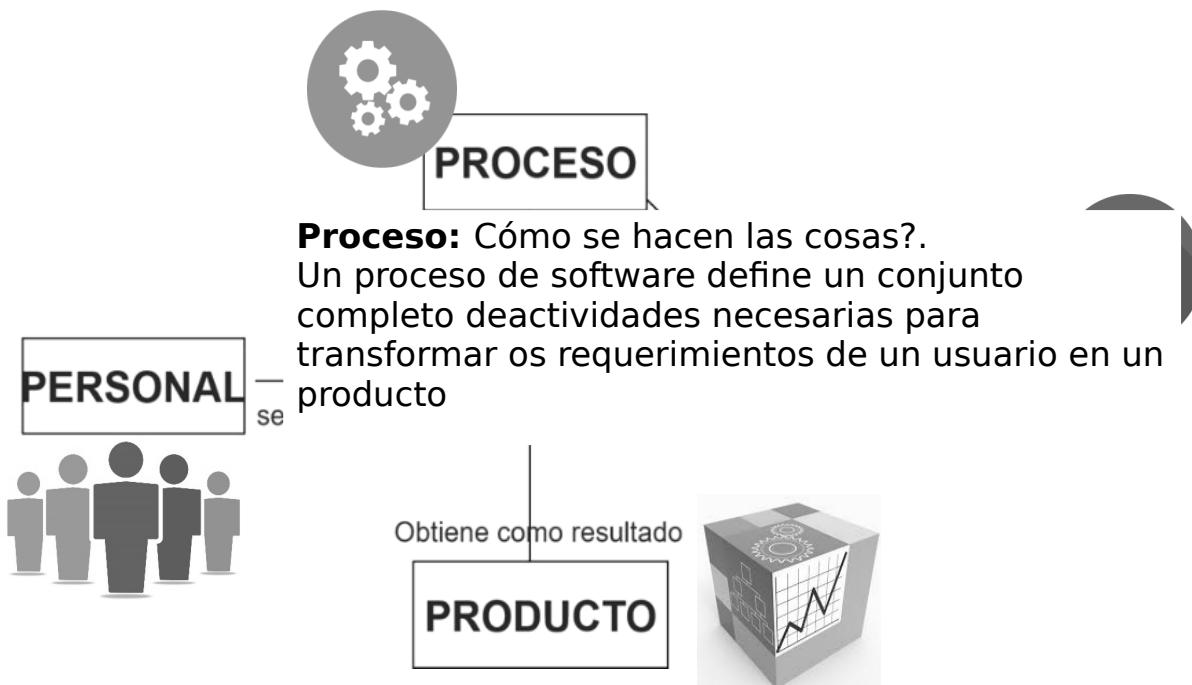
El software como producto se desarrolla en forma gradual, cada versión es única y se obtiene como resultado de la ejecución de un Proyecto



Desarrollo de Software - Las 4 “P”



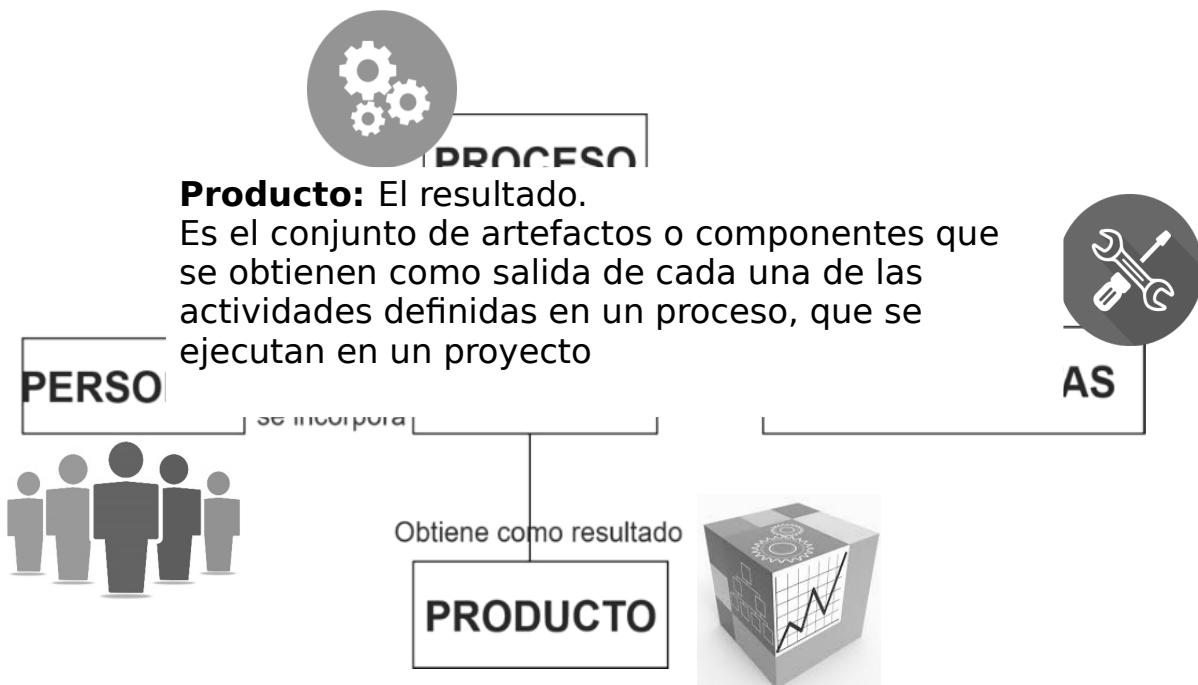
Desarrollo de Software - Las 4 “P”



Desarrollo de Software - Las 4 “P”



Desarrollo de Software - Las 4 “P”



Las 4 “P” - Personas



- Las personas asumen diferentes roles, conformando el **Equipo de Desarrollo**:
 - Analista de sistemas
 - Diseñadores
 - Programadores
 - Arquitectos
 - Analistas de pruebas
 - Líderes de proyectos
 - Revisores técnicos
- Factores para conformar el Equipo de Desarrollo:
 - Experiencia en el dominio de aplicación
 - Experiencia en la plataforma y el lenguaje de programación
 - Habilidad para resolver problemas
 - Habilidad de comunicación
 - Adaptabilidad
 - Actitud
 - Personalidad



Las 4 “P” - Proyecto



- Integra personas, utiliza proceso y herramientas para obtener como resultado un producto de software
- Al principio se debe adaptar el proceso y el ciclo de vida al proyecto
- Un proyecto tiene las siguientes características:
 - **Temporal:** Posee fecha de comienzo y finalización
 - **Productos, servicios o resultados únicos:** Los resultados que se obtienen por similares que sean dos proyectos, tienen características que los hacen únicos
 - **Orientado a objetivos:** Los objetivos deben ser claros, cuando todos comprenden lo que hay que lograr; y alcanzables cuando es factible de hacerse
 - **Elaboración gradual:** Desarrollo en pasos que aumenta mediante incrementos



Las 4 “P” - Producto

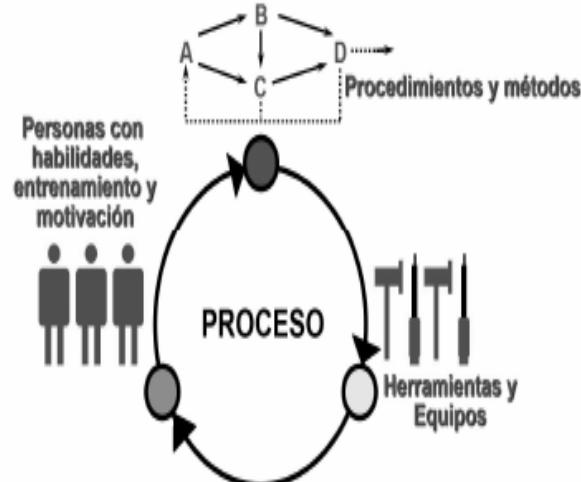


- Se obtiene como consecuencia de una evolución y refinamiento continuo de los modelos
- El producto es más que código, el producto que se obtiene es un sistema de software
- Un **sistema de software** es la sumatoria de todos los artefactos que se necesitan para representarlo en una forma comprensible para las máquinas, los trabajadores y los interesados
- Un **Artefacto** se refiere a cualquier información creada, producida, cambiada o utilizada por las personas en el desarrollo del sistema

Las 4 “P” - Proceso



- Proporciona un marco de trabajo, una estructura que puede tomarse como referencia para definir el plan que guiará el proyecto
- Contiene además de la definición de las actividades, procedimientos y métodos, la identificación de las herramientas que permitan la automatización de las actividades y facilite el trabajo
- El proceso contiene actividades relacionadas con las disciplinas técnicas, de gestión y de soporte o protectoras
- **El Ciclo de Vida del Proceso de Desarrollo de Software** indica las fases/etapas del proceso y el orden en el que se llevarán a cabo



Procesos de Control

Dependiendo de la forma en la que se realizará el trabajo en el contexto del proyecto y de las decisiones que se tomen existen dos procesos de control:

- **Proceso predictivo de control:** Asume que es posible detallar las principales variables de un proyecto (requisitos, tareas, recursos, costos y tiempo) y predecir su comportamiento a largo plazo
- **Proceso empírico de control:** Acepta la complejidad y en vez de pretender controlar la realidad, mide constantemente los resultados y adapta el comportamiento en consecuencia



Procesos de Control - Basado en Línea de Producción

- Asume que podemos repetir el mismo proceso una y otra vez, indefinidamente, y obtener los mismos resultados
- La administración y control provienen de la predictibilidad del proceso definido



Procesos de Control - Empírico

- Asume procesos complicados con variables cambiantes. Cuando se repite el proceso, se pueden llegar a obtener resultados diferentes
- La administración y control es a través de inspecciones frecuentes y adaptaciones
- Son procesos que trabajan bien con procesos creativos y complejos





Ciclo de vida del Proceso de Desarrollo de Software

- Para el diseño y desarrollo de un producto de software se aplican metodologías, modelos y técnicas que permiten resolver problemas
- Se han definido varias metodologías de desarrollo que ayudan a desarrollar software de calidad
- Una metodología define una forma disciplinada para desarrollar software con el objetivo de hacerlo más predecible y eficiente
- Un proceso en particular puede utilizarse con distintos modelos de proceso, en proyectos diferentes, en función de las situaciones de esos proyectos



Modelos de Proceso para el Desarrollo de Software

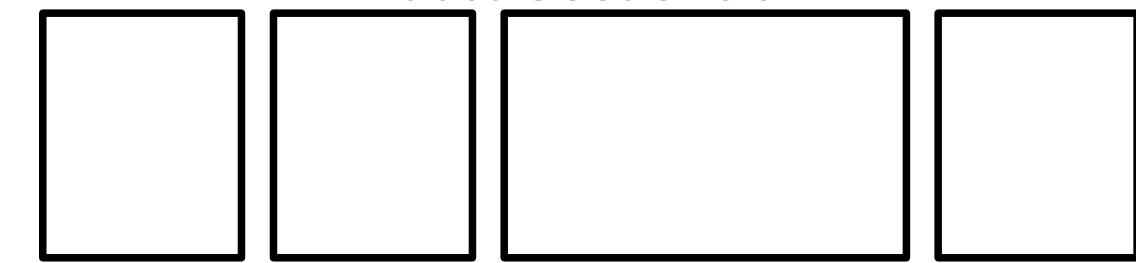
Un *modelo de proceso o ciclo de vida*, es una representación abstracta de un proceso

- Cada modelo representa un proceso desde una perspectiva particular y así proporciona información parcial sobre el proceso
- Los modelos no son descripciones definitivas de los procesos del software, más bien son abstracciones de los procesos que se pueden utilizar para el desarrollo de software
- Hay tres tipos básicos de Ciclos de Vida
 - Secuencial
 - Iterativo/Incremental
 - Recursivo



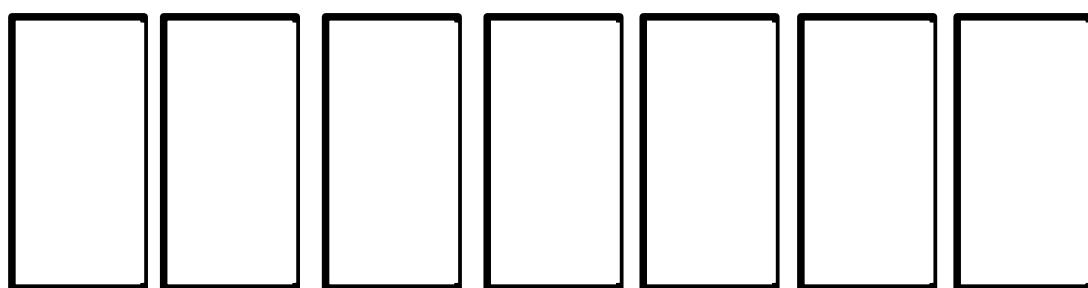
Modelos de Proceso para el Desarrollo de Software

100% Secuencial



Requerimientos Arquitectura Desarrollo Test

100% Iterativo

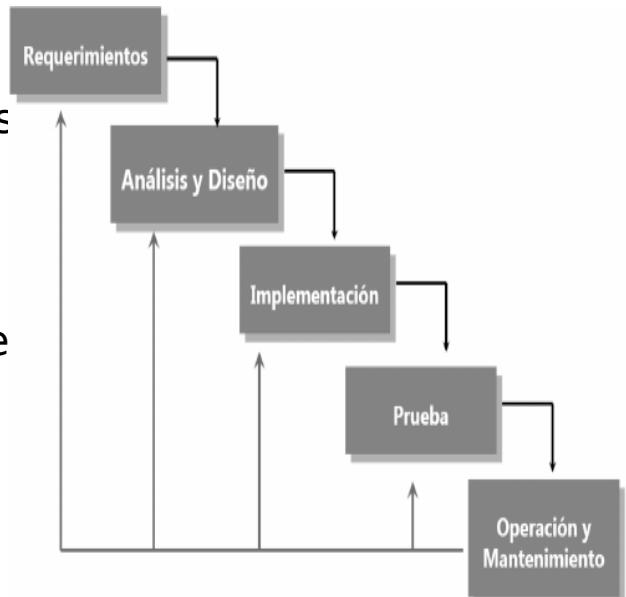


Modelos de Ciclo de Vida

- Build and Fix
- Secuencial
- Cascada
- Cascada con Retroalimentación
- Cascada con Subproyectos
- Modelo V
- Espiral
- Modelo Evolucionario
- RAD (Desarrollo Rápido de Aplicaciones)
- Incremental

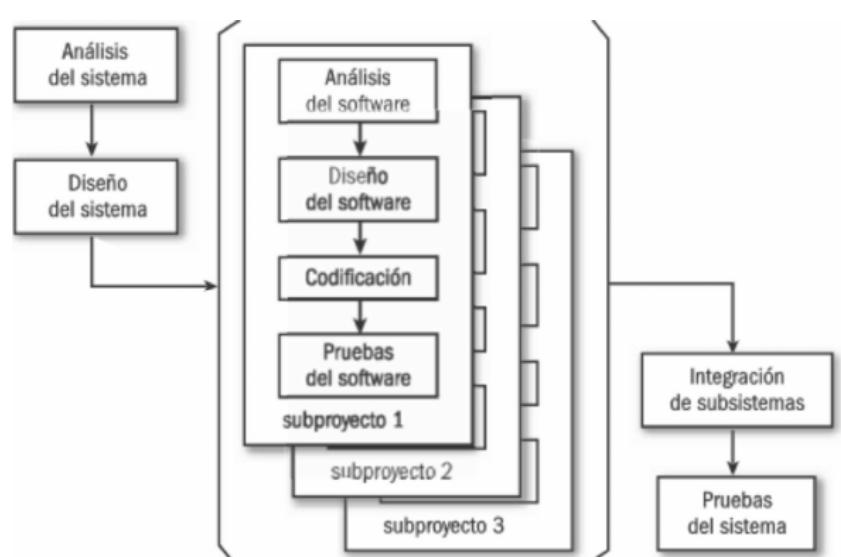
Modelo de Proceso en Cascada

- Considera que las actividades (o etapas) del proceso de desarrollo: Requerimientos, análisis, diseño, implementación y prueba, son etapas separadas que para continuar con la siguiente se debe completar totalmente la anterior
- La retroalimentación con el cliente se realiza una vez que el producto ha sido terminado
- Es muy costoso volver a las etapas anteriores para realizar modificaciones por un malentendimiento de los requerimientos

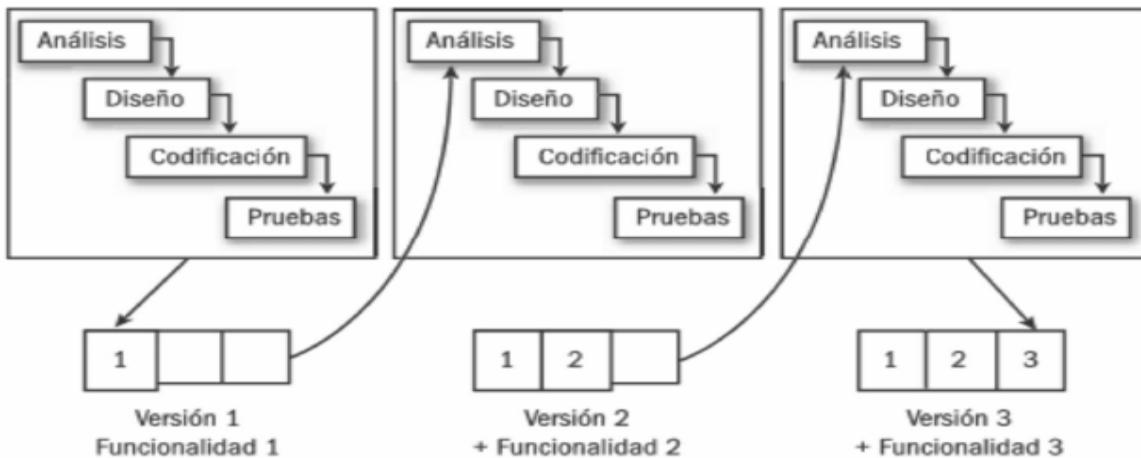


Modelo de Proceso en Cascada con Subproyectos

- Se identifica la funcionalidad al principio y luego se dividen los requerimientos en subproyectos
- En cada subproyecto se realiza el análisis y diseño detallado
- Al final de cada subproyecto se realiza la integración
- Los subproyectos se pueden realizar en paralelo dependiendo de cómo se organiza el equipo de

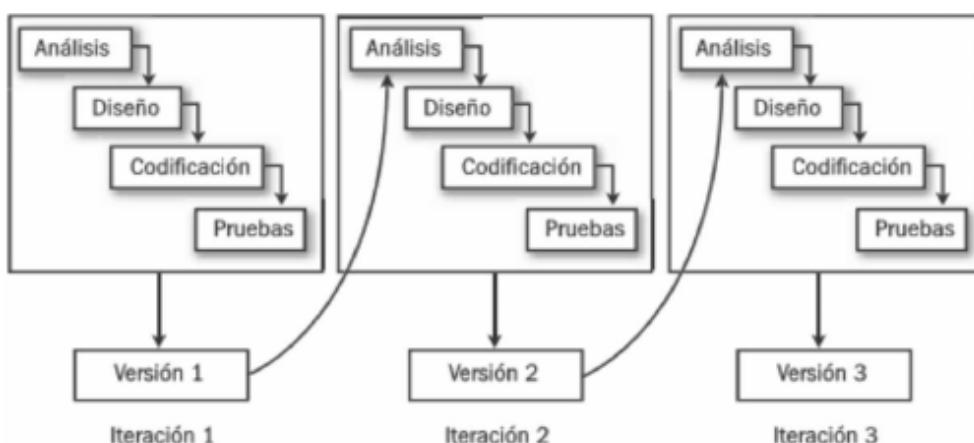


Modelo de Proceso Incremental



- El software se desarrolla gradualmente, por funcionalidades que incrementan el producto
- Se aplican pequeños ciclos de vida en cascada
- Luego de cada ciclo es posible realizar una entrega de software parcial al cliente

Modelo de Proceso Iterativo



- Busca reducir la brecha entre las necesidades del usuario y el producto final
- Después de cada iteración se le entrega al cliente una versión del sistema
- El cliente evalúa el proyecto, lo corrige o propone mejoras
- Se usa cuando los requerimientos no estan claros por parte del cliente



Captura de Requerimientos

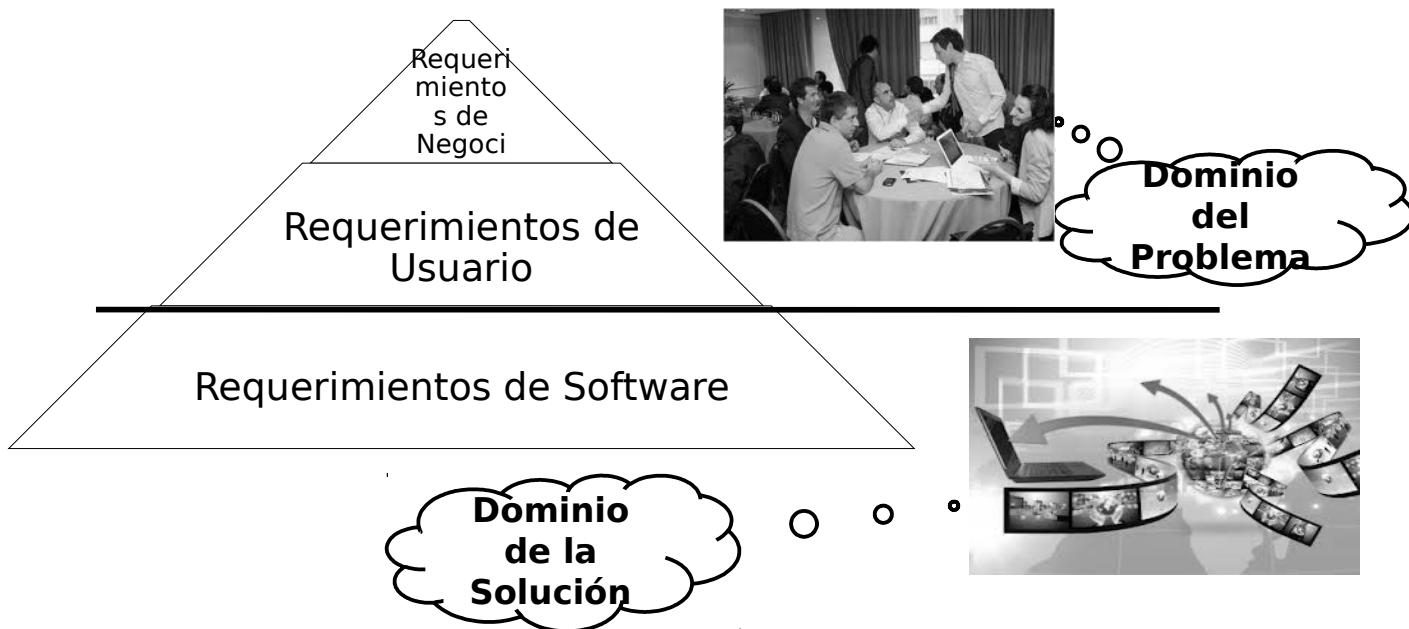
La parte más difícil de construir un sistema de software es decidir precisamente qué construir. Ninguna otra parte del trabajo conceptual, es tan difícil como establecer los requerimientos técnicos detallados.. Ninguna otra parte del trabajo afecta tanto al sistema resultante si se hace incorrectamente. Ninguna otra parte es tan difícil de rectificar más adelante.



Captura de Requerimientos

- La especificación de los requerimientos debe ser clara, sin ambigüedades, en forma consistente y compacta.
- Un requerimiento es una condición o capacidad que debe cumplir el sistema que se desarrollará y que proviene directamente de los usuarios finales. (Libro Proceso Unificado)
- Un requerimiento es una condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo. (IEEE)

Captura de Requerimientos

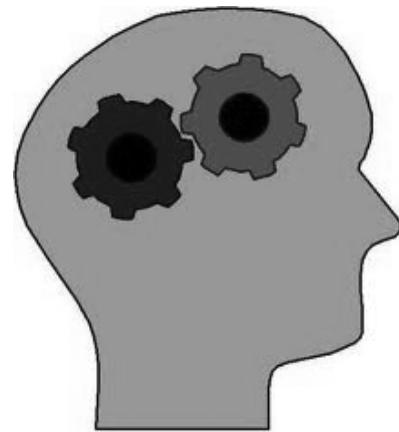


Captura de Requerimientos

- De la definición, selección e implementación de los requerimientos depende el éxito del proyecto
- Los requerimientos son importantes porque permiten:
 - Llegar a un acuerdo entre el cliente y los usuarios de lo que el sistema debería hacer
 - Ayudar al equipo de desarrollo a comprender lo que el sistema debería hacer
 - Delimitar el producto de software a construir
 - Servir de base para la planificación de las iteraciones de un proyecto
 - Definir la interfaz de usuario del sistema

Requerimientos Funcionales

- ◆ Relacionados con la descripción del comportamiento fundamental de los componentes del software
- ◆ Las funciones son especificadas en términos de entradas, procesos y salidas
- ◆ Una vista dinámica podría considerar aspectos como el control, el tiempo de las funciones (de comienzo a fin) y su comportamiento en situaciones excepcionales



Requerimientos No Funcionales

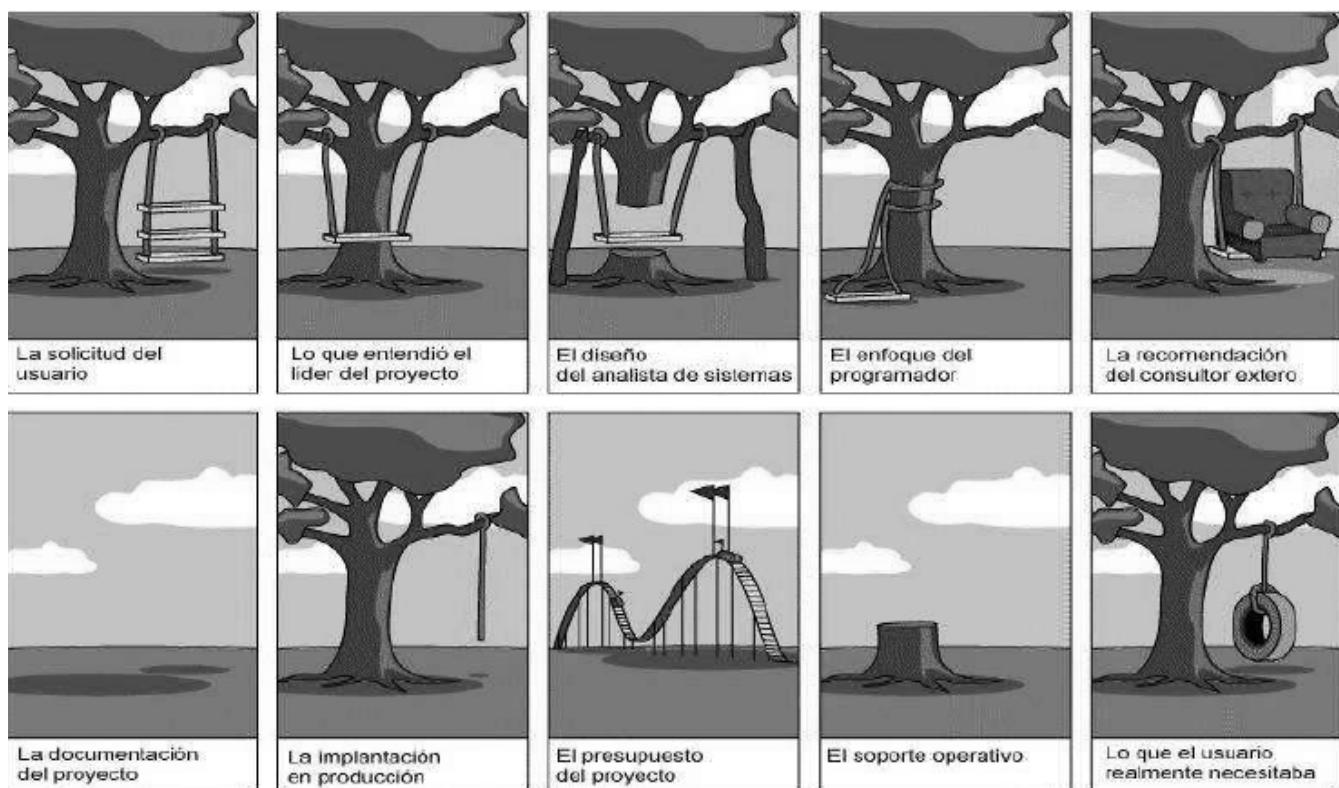
- ◆ **Juegan un papel crucial en el diseño y desarrollo del sistema de información**
- ◆ **Pueden definirse como consideraciones o restricciones asociadas a un servicio del sistema**
- ◆ **Suelen llamarse también requerimientos de calidad o no comportamentales en contraste con los comportamentales**
- ◆ **Pueden ser tan críticos con los funcionales**



Captura de Requerimientos



Captura de Requerimientos - Problemas



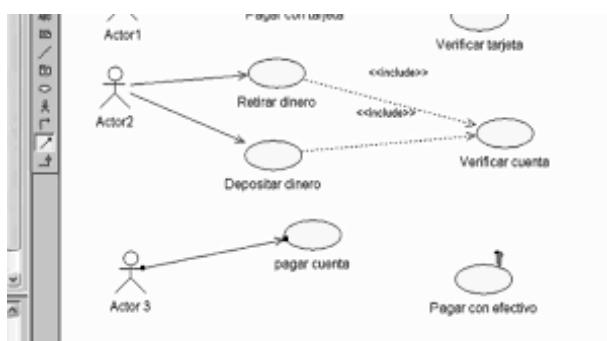
Captura de Requerimientos - Problemas

- Dificultad en el seguimiento de los cambios en los requerimientos
- Dificultad en la especificación de los requerimientos
- Errores en la detección de características esperadas para el software
- Mala organización
- Los requerimientos no son siempre obvios y provienen de fuentes diferentes
- Los requerimientos no son siempre fáciles de expresar de manera claramente mediante palabras
- Existen diferentes tipos de requerimientos a diferentes niveles de detalle
- El número de requerimientos puede volverse inmanejable si no se controla
- Existen varias partes interesadas y responsables, lo que significa que los requerimientos necesitan ser gestionados por grupos de personas de funciones cruzadas
- Los requerimientos cambian ya sea durante el desarrollo del proyecto como una vez que el sistema está funcionando



Captura de Requerimientos - Técnicas de Especificación

Casos de Uso (UML)
(métodos ágiles)



Historias de Usuario



Historia: Ingresar al Blog

Como: Lector del Blog

Quiero: Ingresar al Blog con mi usuario y contraseña

Para: realizar comentarios y mantenerme en contacto con otros lectores que comparten mis intereses

2



Probar el Software - Testing

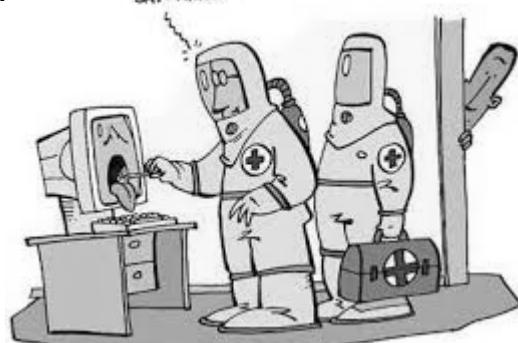
- Pertenece a una actividad denominada Verificación y Validación V&V
- V&V son las actividades que aseguran que el software respeta su especificación y satisface las necesidades de los clientes
- **Validación:** Estamos construyendo el producto correcto?
- **Verificación:** estamos construyendo el producto correctamente?
- La verificación consiste en corroborar que el programa respeta su especificación; y validación significa corroborar que el programa satisface las expectativas del usuario
- Según Dijkstra: “*Si uno de los casos de prueba detecta un error el programa es incorrecto, pero si ninguno de los casos de prueba encuentra un error no podemos decir que el programa es correcto*”



Probar el Software - Testing

- Proceso DESTRUCTIVO de tratar de encontrar defectos en el código.
- Se debe ir con una actitud negativa para demostrar que algo es incorrecto
- El desarrollo exitoso del testing en cuando se encuentran errores NO cuando no se encuentran

SAY: "AHH..."





Conceptos básicos sobre Testing

- Testear un programa significa ejecutarlo bajo condiciones controladas tales que permitan observar su salida o resultado
- **Falta (Failure)**: Cuando un programa funciona mal
- **Falla (Fault)**: Existe en el código del programa. Puede provocar una falta
- **Error**: Acción humana que resulta en software que contiene una falla
- No se puede comprobar que el sistema no tenga una falta, si que esté libre de fallas. El propósito de la prueba es encontrar fallas
- La prueba es un proceso destructivo, debe buscar qué cosas se hicieron mal
- La corrección de una falla puede provocar fallas adicionales



Principios del Testing de Software

- Un programador debería evitar probar su propio código
- Una unidad de programación no debería probar sus propios desarrollos
- Examinar el software para probar que no hace lo que se supone que debería hacer
- Examinar el software para detectar que hace lo que no se supone que debería hacer
- No planificar el esfuerzo de testing sobre la suposición de que no se van a encontrar defectos
- El testing es una tarea extremadamente creativa e intelectual desafiante
- Testing temprano, lo más temprano posible
- La paradora del pesticida:
- El testing es dependiente del contexto
- Falacia sobre la ausencia de errores



Tipos de Pruebas

• Testing Funcional

- Las pruebas se basan en funciones y características (descripta en los documentos o entendidas por los analistas de prueba) y su interoperabilidad con sistemas específicos
 - Basado en Requerimientos
 - Basado en los procesos de negocio



Tipos de Pruebas

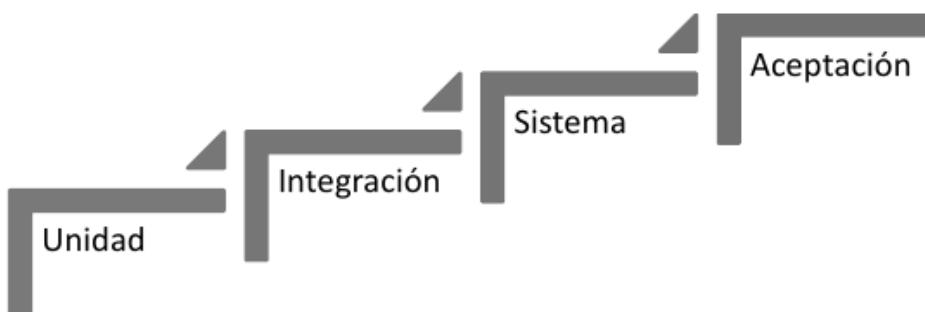
Testing No Funcional

Es la prueba de “cómo” funciona el sistema

- Los requerimientos no funcionales son tan importantes como los funcionales
 - Performance Testing
 - Pruebas de Carga
 - Pruebas de Stress
 - Pruebas de Usabilidad
 - Pruebas de Mantenimiento
 - Pruebas de Fiabilidad
 - Pruebas de Portabilidad

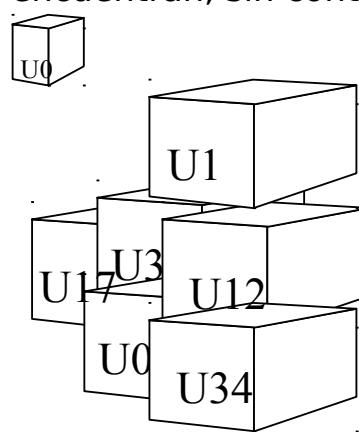
Tipos de Prueba

- Las pruebas (tests) se hacen para verificar y validar tanto requerimientos funcionales como requerimientos no funcionales
- Existen diferentes niveles de test
 - Test de Unidad
 - Test de Integración
 - Test del Sistema
 - Test de Aceptación



Test de Unidad

- Se prueba una y solo una componente (case, bloque)
- Normalmente lo realiza el desarrollador
- Cada componente es probada independientemente
- Se produce con acceso al código bajo pruebas y con el apoyo del entorno de desarrollo, tales como framework de pruebas unitarias o herramientas de depuración
- Los errores se suelen reparar tan pronto como se encuentran, sin contención oficial del incidente





Test de Integración

- Se verifica si las unidades trabajan juntas correctamente (casos de uso, subsistemas)
- Los módulos críticos deben ser probados lo más tempranamente posible
- Los puntos clases del test son simples:
 - Conectar de a poco las partes más complejas
 - Minimizar la necesidad de programas auxiliares
- Son necesarias porque:
 - Al combinar las unidades pueden aparecer nuevas fallas
 - La combinación aumenta exponencialmente el número de caminos posibles



Test del Sistema

- Se testea el sistema completo
- Trata de determinar si el sistema en su globalidad opera satisfactoriamente
- El entorno de prueba debe corresponder al entorno de producción tanto como sea posible para reducir al mínimo el riesgo de incidentes debidos al ambiente específicamente y que no se encontraron en las pruebas.
- Deben investigar tanto requerimientos funcionales y no funcionales del sistema.



Técnicas de Testing

- **Test de regresión**
- **Test de operación**
- **Test de escala completa**
- **Test de performance o capacidad**
- **Test de sobrecarga**
- **Test negativo**
- **Test ergonomico**
- **Test de la documentación del usuario**



Test de Regresión

Se realiza cuando se han hecho modificaciones al sistema.

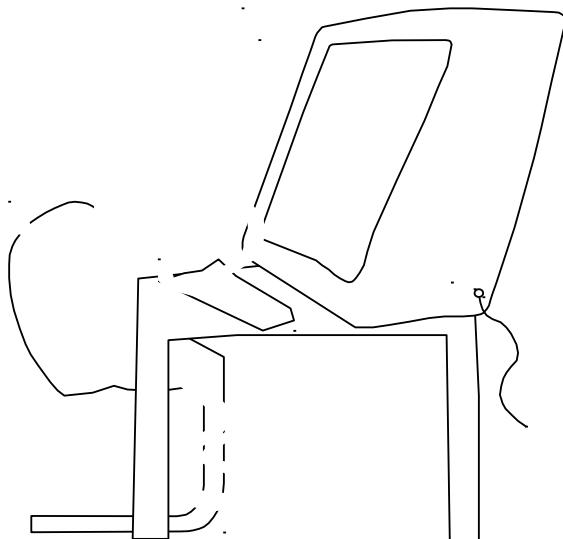
El objetivo es verificar que la funcionalidad anterior se mantenga



Test de Operación

Es el más común

El sistema es probado en operaciones normales



Test de Escala Completa

**El sistema se ejecuta con todos los
valores al máximo
Todos los parámetros del sistema en los
valores límites, muchos equipos
conectados y el sistema usado por varios
usuarios**



Test de Performance o Capacidad

Mide la capacidad de procesamiento del sistema (CPU, velocidad)
El test es diseñado para medir la performance con diferentes cargas



Test de Sobrecarga

Va un paso más de los de performance y escala completa
El objetivo es ver cómo se comporta el sistema cuando
tiene sobrecarga





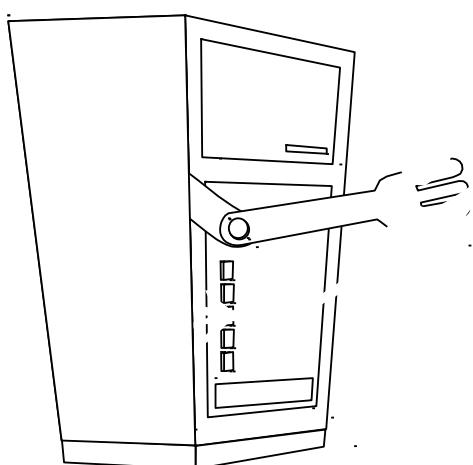
Test Negativo

Es una clase de escala total donde todos los parámetros son puestos en valores fuera de lo permitido



Test Ergonómico

Si el sistema será usado por usuarios inexpertos evalúa cuán intuitiva es la interfaz

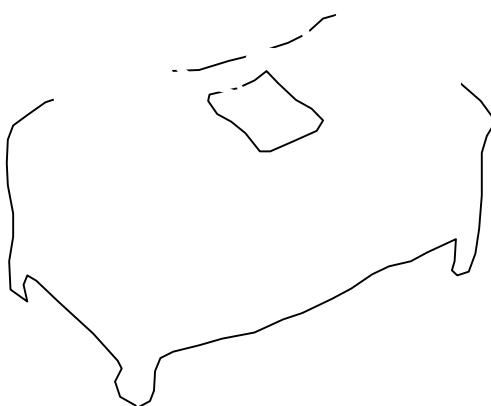




Test de Documentación

La documentación del sistema es testeada

Tanto el manual del usuario como la documentación son testeadas para analizar si las decisiones más importantes del sistema están claramente justicadas

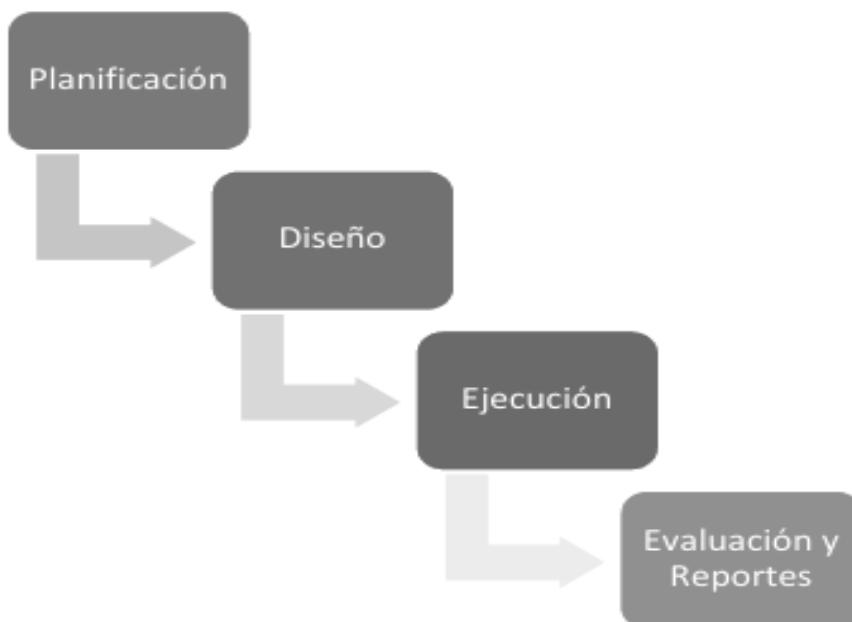


Test de Aceptación

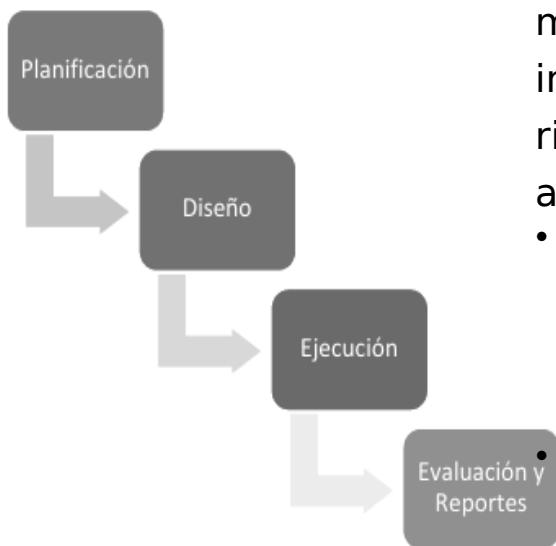
El cliente está de conforme con el sistema entregado y la capacitación realizada

Proceso de Prueba

- Es importante planear una prueba
- El proceso de prueba corre en paralelo con otros procesos



Proceso de Prueba - Planificación



- La Planificación de las pruebas es la actividad de verificar que se entienden las metas y los objetivos del cliente, las partes interesadas (stakeholders), el proyecto, y los riesgos de las pruebas que se pretende abordar
 - Construcción del Test Plan:
 - Riesgos y Objetivos de la prueba
 - Estrategia de prueba
 - Recursos
 - Criterio de Aceptación
 - Controlar:
 - Revisar los resultados de la prueba
 - Cobertura y criterio de aceptación

Proceso de Prueba - Diseño



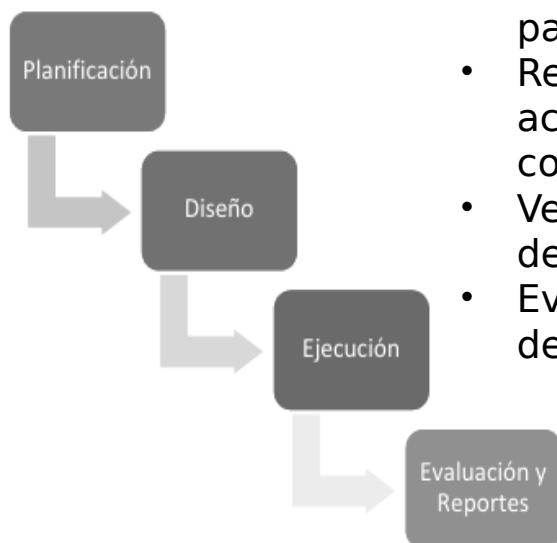
- Revisión de la base de pruebas
 - Verificación de las especificaciones para el software bajo pruebas
 - Evaluar la verificabilidad de los requerimientos y el sistema
 - Identificar los datos necesarios
 - Diseño y priorización de los casos de las pruebas
 - Diseño del entorno de prueba

Proceso de Prueba - Ejecución



- Desarrollar y dar prioridad a nuestros casos de prueba
 - Crear los datos de prueba
 - Automatizar lo que sea necesario
 - Creación de conjuntos de pruebas de los casos de prueba para la ejecución de la prueba eficientemente
 - Implementar y verificar el ambiente
 - Ejecutar los casos de prueba
 - Registrar el resultado de la ejecución de pruebas y registrar la identidad y las versiones del software en las herramientas de pruebas.
 - Comparar los resultados reales con los resultados esperados

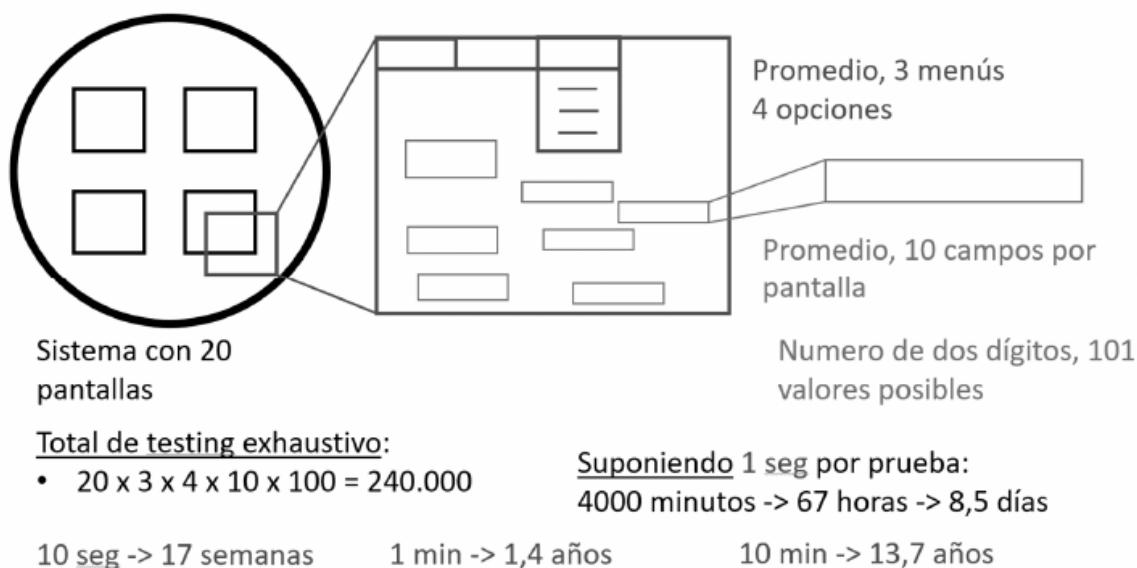
Proceso de Prueba - Evaluación y Reportes



- Evaluar los criterios de Aceptación
- Reporte de los resultados de las pruebas para los stakeholders
- Recolección de la información de las actividades de prueba completadas para consolidar.
- Verificación de los entregables y que los defectos hayan sido corregidos
- Evaluación de cómo resultaron las actividades de testing y se analizan las lecciones aprendidas

Cuánto testing es suficiente?

- Decidir que se ha testeado lo suficiente depende de:
 - Evaluación del nivel de riesgo
 - Costos asociados al proyecto





Cuándo dejar de probar?

- Es casi imposible estimar cuándo se justifique finalizar el proceso de prueba
- Algunas estrategias utilizadas son:
 - Dejar de probar cuando el producto para exitosamente el conjunto de pruebas diseñado
 - “Good enough”, cierta cantidad de fallas no críticas es aceptable
 - Defectos detectados es similar a la cantidad de defectos estimados
- Encontrar y corregir defectos, no quiere decir que se construyó el sistema que el cliente quería
- La gente que usa software está más interesada en que la aplicación les sirva para resolver sus problemas diarios
- Las revisiones en las primeras etapas del ciclo de vida son una parte importante de las pruebas ya que los usuarios pueden transmitir lo que realmente necesitan