Universidad Autónoma del Estado de México

Centro Universitario UAEM Valle de Chalco

PROGRAMA EDUCATIVO

INGENIERÍA EN COMPUTACIÓN

UNIDAD DE APRENDIZAJE

TEORÍA DE SISTEMAS

MATERIAL:

DESARROLLO DE SOFTWARE

ELABORADO POR:

DRA. ANABELEM SOBERANES MARTÍN

2016

TEORÍA DE SISTEMAS

Objetivo:

Que el estudiante obtenga una visión general de la ingeniería de software, así como de los principales modelos y ciclos de vida del software y de las metodologías asociadas a ellos, que sirva de soporte a los cursos de análisis y de diseño, mediante el estudio y seguimiento de una metodología en sus fases iniciales, sustentado en la Teoría General de Sistemas.

Presentación

- Las organizaciones han reconocido la importancia de una administración adecuada de los recursos básicos, tales como la mano de obra y las materias primas; sin embargo, ahora la información ha adquirido una connotación de recurso primordial.
- La información puede llegar a ser el elemento decisivo, que en un momento dado, determine el éxito o el fracaso de un negocio, es por esto que el estudio de la Ingeniería de software queda incompleto sin reflexionar que los sistemas de información deben sustentarse o desarrollarse a partir de las necesidades de la organización a la cual deben de servir.

ESTRUCTURA DE LA UNIDAD DE APRENDIZAJE

UNIDAD DE COMPETENCIA I.

- Introducción a la ingeniería de software.

UNIDAD DE COMPETENCIA II.

- Componentes de un sistema de información.
- Enfoque de sistemas

ESTRUCTURA DE LA UNIDAD DE APRENDIZAJE

UNIDAD DE COMPETENCIA III.

- Integrar los sistemas de información dentro de una organización.

UNIDAD DE COMPETENCIA IV.

- Diferenciar los distintos tipos de sistemas de información computarizados y sus elementos principales.

ESTRUCTURA DE LA UNIDAD DE APRENDIZAJE

UNIDAD DE COMPETENCIA V.

- Analizar las diferentes metodologías para el desarrollo de sistemas de información.

Temas de las Unidades de competencia abordados en la presentación

UNIDAD DE COMPETENCIA IV

Ingeniería de software.

UNIDAD DE COMPETENCIA V

Etapas, elementos, ventajas, desventajas.
Modelos.

CONTENIDO

I. INGENIERÍA DE SOFTWARE

Fritz Bauer

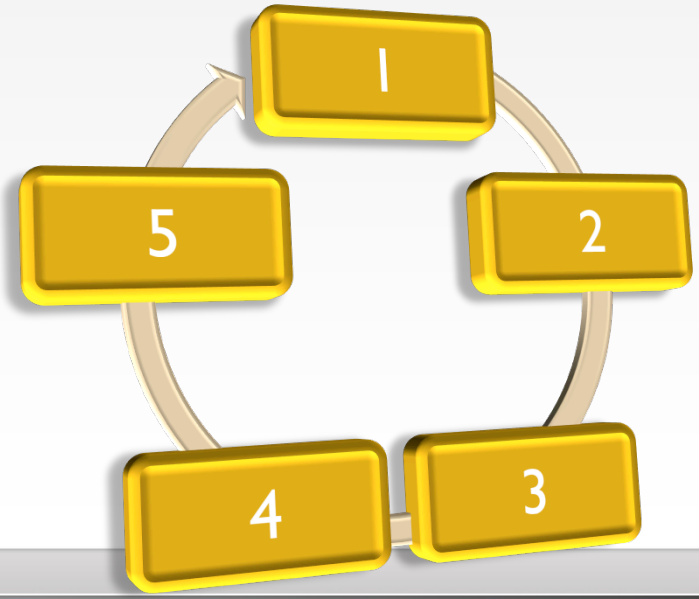
La ingeniería de sw es el establecimiento y uso de principios sólidos de ingeniería para obtener económicamente un sw confiable y que funcione de modo eficiente en máquinas reales.

IEEE define a la ingeniería de software, como:

La aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento del software, es decir la aplicación de la ingeniería de sw.

II. PROCESO DEL SOFTWARE

- Proceso: Define quién hace qué, cuándo y cómo para alcanzar cierto objetivo. En general, el éxito de las empresas u organizaciones depende en gran medida de la definición y seguimiento adecuados de sus procesos.



- El proceso de desarrollo de software se puede definir como el conjunto de actividades, métodos, prácticas y transformaciones que los individuos emplean para desarrollar y mantener el software, así como los productos asociados (Paul, M. C., Weber, C. V., Curtis, B. y Chris, M. B., 1995).

Existen cuatro actividades fundamentales de procesos que son comunes para todos los procesos del software:

① Especificación del Software
② Desarrollo del Software
③ Validación del Software
④ Evolución del Software

① Especificación del Software

- Es donde los clientes e ingenieros definen el software a producir y las restricciones sobre su operación.

Item	Instruction
Processor	Special Dual Cell Processor for automotive
Flash	512KB
Wired socket	USB2.0/USB1.1 Compatible
Diagnostic Socket	24PIN Adapter(Self-determination design)
Wireless Socket	802.11b/g Wireless WIFI
Indicator Light	4 LED indicator lights，Indicate Power and operation state
Power	DC 9V-28V Vehicle's power input
Consumption	3W
Size	LxWxH=156x86x35(mm)
Working Temperature	-20～+70 °C
Storage Temperature	-40～+85 °C

② Desarrollo del Software

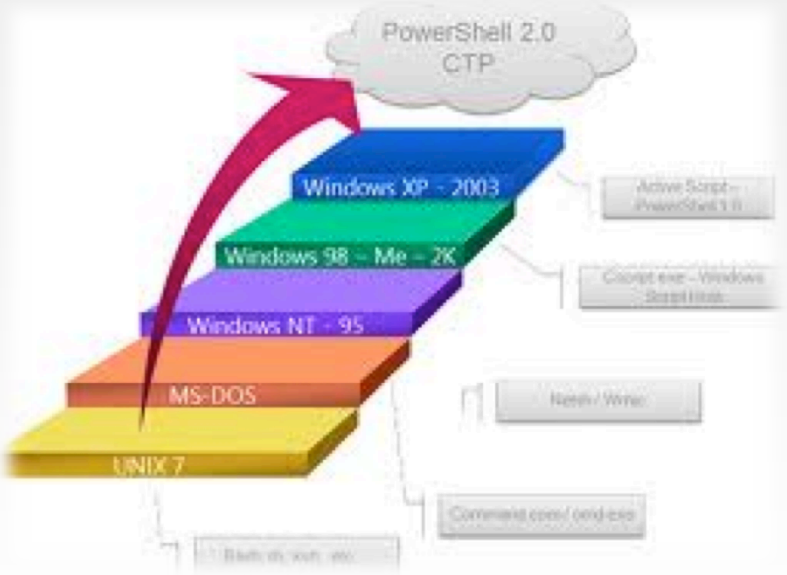
- Etapa en donde el software se diseña y se programa.

③ Validación del Software

- Donde el software se valida para asegurar que es lo que el cliente requiere.

④ Evolución del Software

- Donde el software se modifica para adaptarlo a los cambios requeridos por el cliente y el mercado.



III. MODELO DE PROCESOS

- Un modelo de Procesos de software define cómo solucionar la problemática del desarrollo de sistemas de software. Para desarrollar el software se requiere resolver ciertas fases de su proceso, las cuales se conocen en su conjunto como el ciclo de vida del desarrollo de software.

Arquitectura

- Una arquitectura de software define la estructura general de un sistema y varia de acuerdo con el tipo de sistema a desarrollar.

Elemento	Probabilidad
Interfaces	Alto
Funcionalidad	Alto
Datos	Medio
Funciones	Medio
Objetos	Bajo
Información	Bajo

Actividad

Una actividad es una unidad o paso básico de un proceso. En el proceso de software las actividades definen los pasos necesarios para lograr las metas y los objetivos; por ejemplo, especificar los requisitos del sistema.

Las actividades deben:

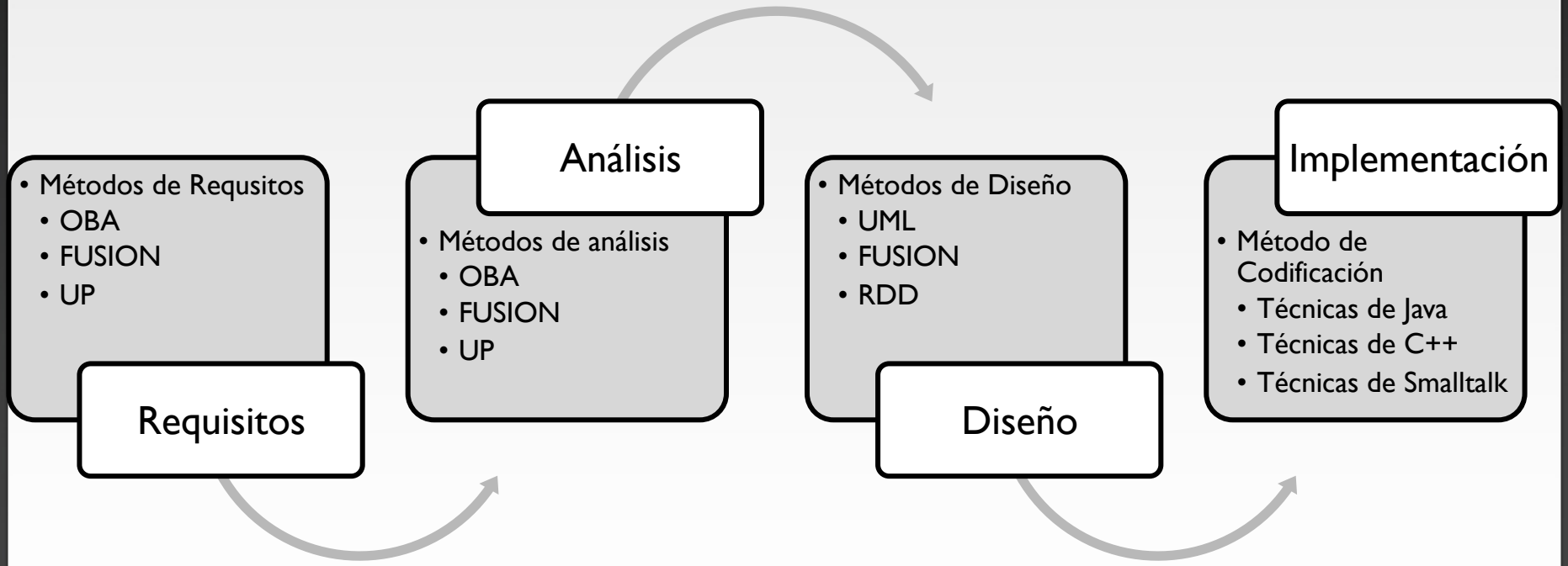
- Ser fáciles de definir y seguir,
- Simplificar la comprensión del sistema y,
- Ofrecer flexibilidad, precisión y extensibilidad.

Actividades del desarrollo de software

IV. MÉTODOS Y METODOLOGÍAS

- Los métodos definen las reglas para las transformaciones internas de las actividades mientras que las metodologías definen el conjunto de métodos.

Contraste de las actividades y métodos de desarrollo de software



Estrategias

- Una estrategia se define como un plan para lograr un objetivo. Las estrategias afectan aspectos como la arquitectura del sistema, el orden en que se llevarán a cabo las actividades del proceso y las metodologías a utilizarse. Otras estrategias aceptadas en la actualidad son los prototipos y la reutilización.

Prototipos

- Un prototipo es una versión preliminar, intencionalmente incompleta o reducida de un sistema.

Tipos de Prototipos

Reutilización

- Es la explotación de componentes desarrollados anteriormente dentro de un mismo proyecto o entre proyectos. La reutilización entre proyectos requiere planeación y representa una inversión tanto para producir componentes reutilizables como para consumir.

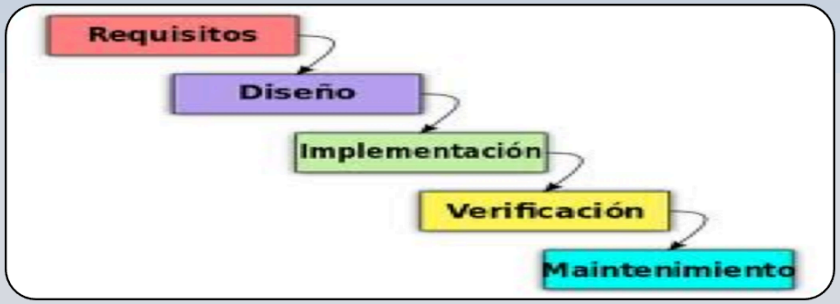
Herramientas

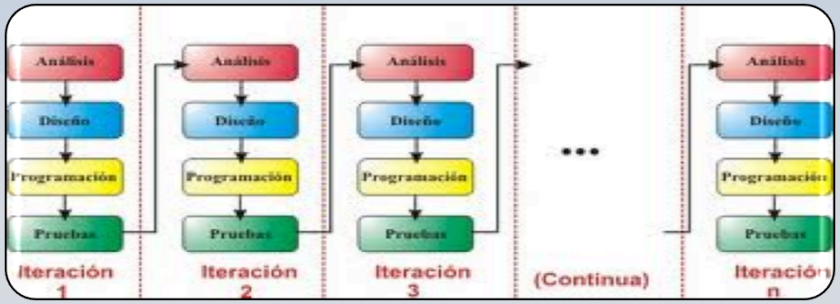
- Las herramientas son aplicaciones que apoyan la administración del proceso de software. Las herramientas CASE, son indispensables en la administración del proceso de software.

V. MODELOS CLÁSICOS

- Los modelos de procesos dependen de las opiniones o creencias de las personas involucradas en un proyecto.
- Por ejemplo:
  - Es necesario comprender el problema antes de desarrollar una solución.

Modelos Clásicos





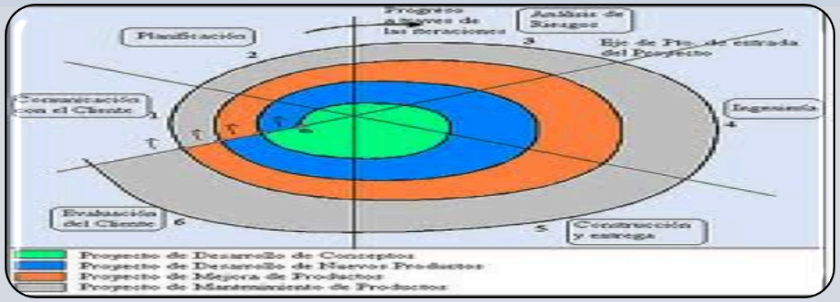
Cascada

- Se define como una secuencia de actividades.

Incremental

- Cada incremento tiene su propio ciclo de vida.

Modelos Clásicos



Evolucionario

- Es una extensión al modelo incremental.

Espiral

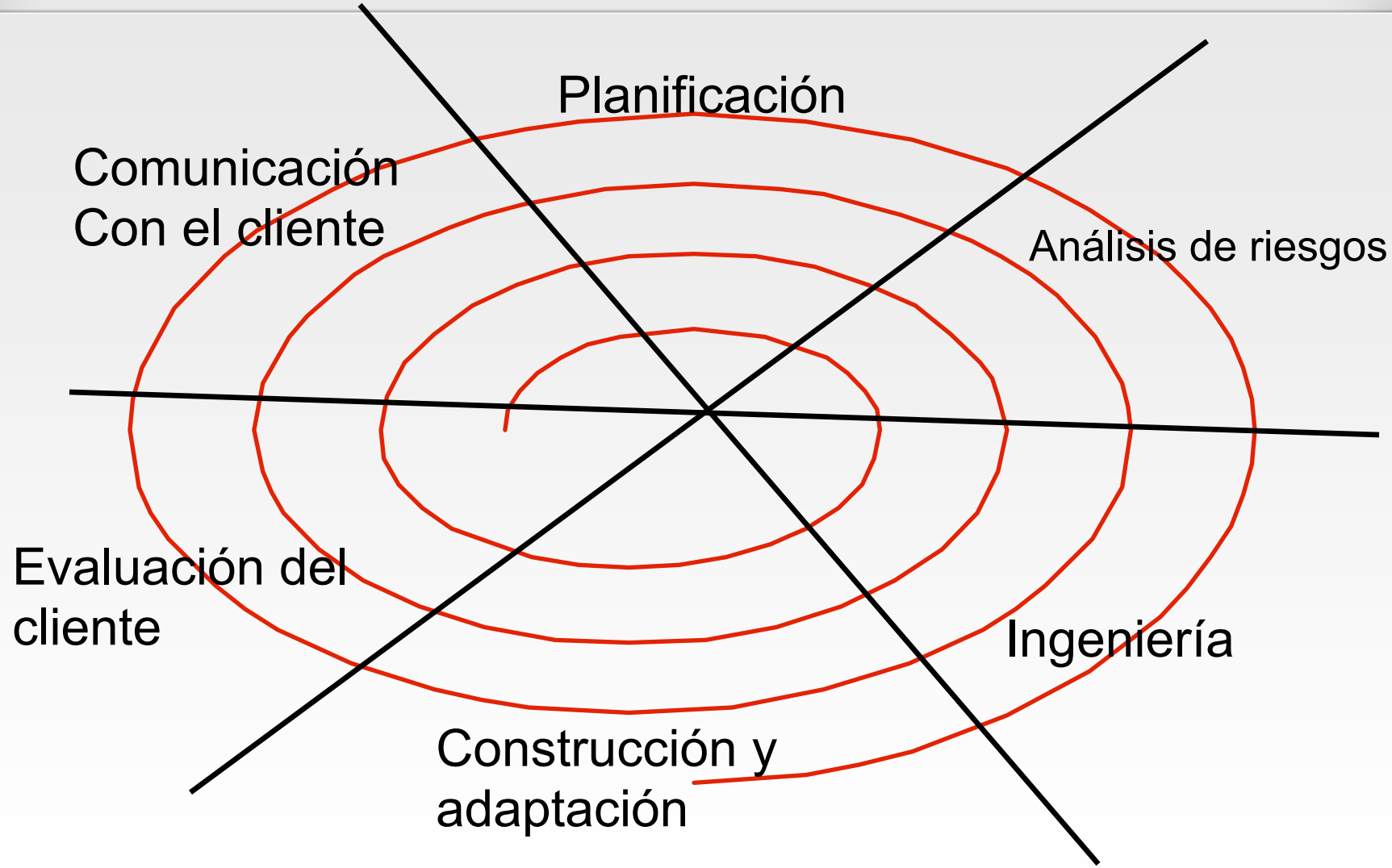
Es una extensión del modelo de cascada

Modelo en espiral

Fue propuesto originalmente por Boehm, es un modelo de proceso de software evolutivo que acompaña la naturaleza interactiva de construcción de prototipos con los aspectos controlados y sistemáticos del modelo lineal secuencial.

- En el modelo espiral, el software se desarrolla en una serie de versiones incrementales.
- En las primeras iteraciones, la versión incremental podría ser un modelo en papel o un prototipo.
- En las últimas iteraciones, se producen versiones cada vez mas completas de ingeniería del sistema.

Actividades estructurales



Un modelo en espiral típico.

Ventajas del Modelo de Espiral

- Centra su atención en la reutilización de componentes y eliminación de errores en información descubierta en fases iníciales.
- Los objetivos de calidad son el primer objetivo.
- Integra desarrollo con mantenimiento.

Problemas con el Modelo de Espiral

- El desarrollo contractual especifica el modelo del proceso y los resultados a entregar por adelantado.
- Requiere de experiencia en la identificación de riesgos.
- Es difícil de convencer al cliente de que el enfoque evolutivo es controlable.

VI. DESARROLLO ÁGIL DE SOFTWARE

- Se entiende como Desarrollo ágil de Software a un paradigma de Desarrollo de Software basado en procesos ágiles.
- Los procesos ágiles de desarrollo de software, conocidos anteriormente como metodologías livianas, intentan evitar los tortuosos y burocráticos caminos de las metodologías tradicionales enfocándose en la gente y los resultados

AGILIDAD

Es dinámica, con contenido específico, ajustable al cambio de manera dinámica y orientada al crecimiento.

LA ALIANZA ÁGIL DEFINE 12 PRINCIPIOS PARA ALCANZAR LA AGILIDAD

1

Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software valioso.

2

Bienvenidos los requisitos cambiantes, incluso en fases tardías del desarrollo. La estructura de los procesos ágiles cambia para la ventaja competitiva del cliente.

3

Entrega con frecuencia software en funcionamiento, desde un par de semanas hasta un par de meses, con una preferencia por la escala de tiempo más corta.

4

La gente de negocios y los desarrolladores deben trabajar juntos a diario a lo largo del proyecto.

5

Construir proyectos alrededor de individuos motivados. Darles el ambiente y el soporte que necesitan, y confiar en ellos para obtener el trabajo realizado.

6

El método más eficiente y efectivo de transmitir información hacia y dentro de un equipo de desarrollo es la conversación cara a cara.

7

El software en funcionamiento es la medida primaria del progreso.

8

Los procesos ágiles promueven el desarrollo sustentable. Los patrocinadores, desarrolladores y usuarios deber ser capaces de mantener un paso constante de manera indefinida.

9

La atención continua a la excelencia técnica y al buen diseño mejora la agilidad.

10

La simplicidad – el arte de maximizar la cantidad de trabajo no realizado – es esencial.

11

Las mejores arquitecturas, los mejores requisitos y los mejores diseños emergen de equipos auto-organizados.

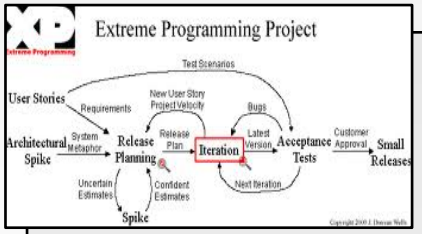
12

A intervalos regulares el equipo refleja la forma en que se puede volver más efectivo; entonces su comportamiento se ajusta y adecua en concordancia.

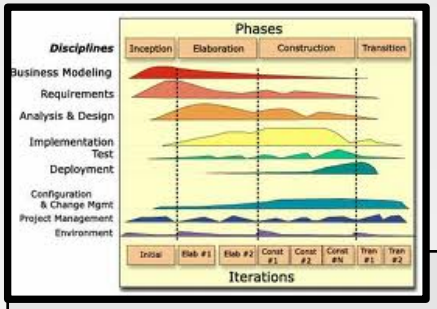
Modelos recientes

- Los modelos de procesos recientes son los siguientes:
  - Ganar-ganar
  - Programa extrema (XP)
  - Proceso unificado (UP)

Ganar-ganar, (win-win) extiende el modelo espiral, haciendo énfasis en la identificación de las condiciones de ganancia para todas las partes, creando un plan para alcanzar las condiciones ganadoras.



Programación extrema, (XP, eXtreme Programming) es un modelo de procesos de software que toma los principios y prácticas aceptadas, y las lleva a niveles extremos.

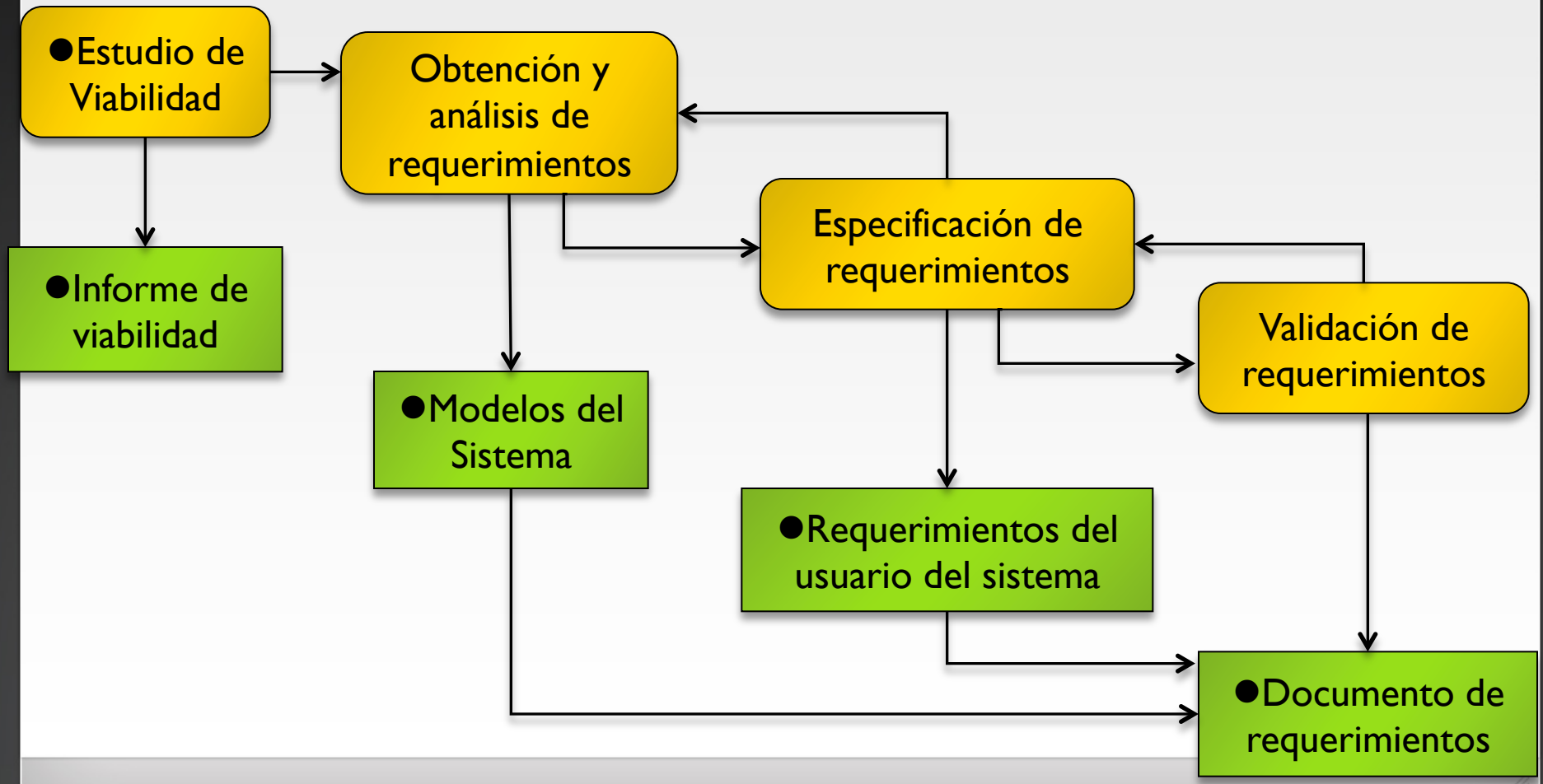


Proceso Unificado (UP, Unified Process) se basa principalmente en la especificación de requerimientos de un sistema mediante casos de uso.

Procesos de Ingeniería de Requerimientos

- La meta del proceso de ingeniería de requerimientos es crear y mantener un documento de requerimientos del sistema.

El proceso de ingeniería de requerimientos



Estudios de viabilidad

- Para todos los sistemas nuevos, el proceso de ingeniería de requerimientos debería empezar con un estudio de viabilidad.
- Un estudio de viabilidad es un estudio corto y orientado a resolver cuestiones:
  - ¿Contribuye el sistema a los objetivos generales de la organización?
  - Se puede implementar el sistema utilizando la tecnología actual y dentro de las restricciones de coste y tiempo?
  - Puede integrarse el sistema con otros sistemas existentes en la organización?

Obtención y Análisis de Requerimientos

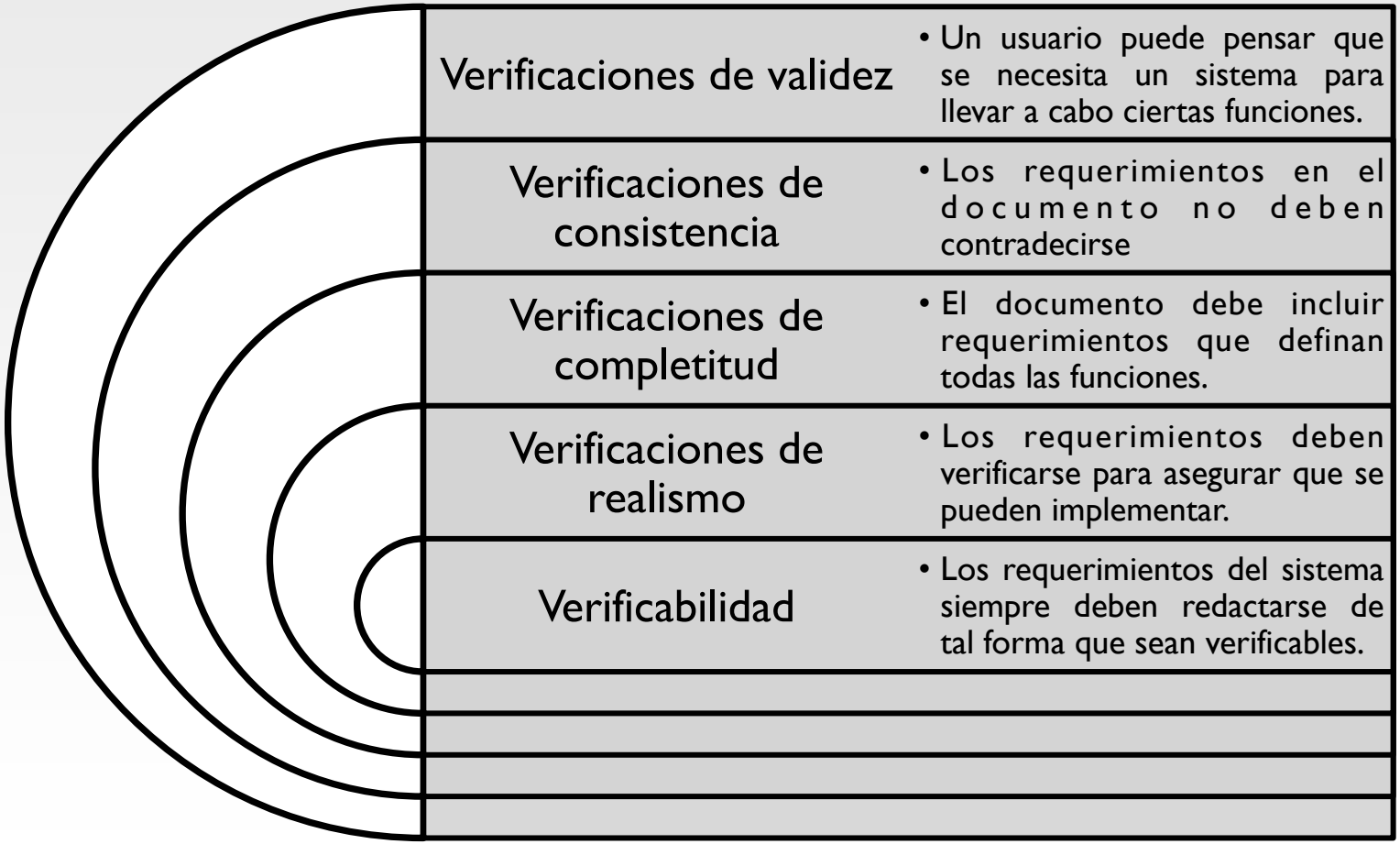
- Es la siguiente etapa de procesos de ingeniería de requerimientos, en ella, los ingenieros de software trabajan con los clientes y los usuarios finales del sistema para determinar el dominio de la aplicación, qué servicios debe proporcionar el sistema, el rendimiento requerido del sistema, las restricciones de hardware, etc.

Obtención y análisis de requerimientos

Validación de Requerimientos

- La validación de requerimientos trata de mostrar que éstos realmente definen el sistema que el cliente desea. Coincide parcialmente con el análisis ya que éste implica encontrar problemas con los requerimientos.
- Se deben llevar a cabo verificaciones sobre requerimientos en el documento de requerimientos.

Estas verificaciones que comprende la validación de requerimientos



Gestión de Requerimientos

- Los sistemas para sistemas software grandes son siempre cambiantes. Una razón es que estos sistemas normalmente se desarrollan para abordar problemas.
- La gestión de requerimientos es el procesos de comprender y controlarlos cambios en los requerimientos del sistema.

Gestión de Requerimientos

- Requerimientos duraderos y volátiles
- Planificación de la gestión de requerimientos
- Gestión del cambio de los requerimientos

Marco de Trabajo

- Establece las bases para un proceso de software completo al identificar un número pequeño de actividades aplicables a todos los proyectos de software, sin importar su tamaño o complejidad.

Marco de Trabajo Genérico

Comunicación,

- Con los clientes, investigación de requisitos.

Planeación,

- Plan para el trabajo de la ingeniería de sw. Describe las tareas técnicas, los riesgos probables, los recursos y un programa de trabajo.

Modelado,

- Creación de modelos que permita al desarrollador y al cliente entender mejor los requisitos del sw y el diseño que logrará satisfacerlos.

Construcción,

- Generación del código y las pruebas necesarias.

Despliegue,

- El sw se entrega al cliente, quien evaluará el producto y dará información sobre la evaluación.

VII. REFERENCIAS DE CONSULTA

Bruegge, B. (2002). Ingeniería de Software. Orientada a Objetos. Prentice Hall.

Fairley, R. (1988). Ingeniería de software. McGraw Hill,

García, G. A., & Rodríguez, J. (2006). *Aplicación del modelado de procesos en un curso de ingeniería de software.* México: Red Revista Electrónica de Investigación Educativa.

Lawrence, S. (2002). Ingeniería de Software, Teoría y Práctica, Argentina: Prentice Hall.

Pressman Roger S. (2004). Ingeniería de Software. Un enfoque práctico, 5ta. Edición, McGraw Hill.

Sommerville, I. (2005). *Ingeniería del Software.* Madrid: Pearson Educación.

Weitzenfeld, A. (2005). *Ingeniería de Software Orientada a Objetos con UML, Java e Internet.* México: Thomson.