Resumen Metodología de Sistemas I

Sistema de información

Sistema de info para Andrew: conjunto formal de procesos que, operando sobre una colección de datos estructurada según las necesidades de la empresa, recopilan, elaboran y distribuyen la información (o parte de ella) necesaria para las operaciones de dicha empresa y para las actividades de dirección y control correspondientes para desempeñar su actividad de acuerdo a su estrategia de negocio.

En resumen, Un sistema de información es un conjunto de datos que interactúan entre sí con un fin común.

Objetivo: ayudar al desempeño de actividades en todos los niveles de la organización, mediante suministro de información de calidad a la persona indicada en el momento y lugar oportuno y con el formato más útil.

El valor de sistema: su eficacia, su extensión, su aceptación por parte de los que lo utilizan, coste (mano de obra optimizando tiempos, tareas, rutinas, etc.), calidad de info que trate y produce (Se entiende a la calidad como un sistema de control o de supervisión para corregir errores), etc.

Concepto de información

Dentro de un sistema de información encontramos:
* Datos: conjunto de señales o signos
* Comunicación: proceso mediante el cual un conjunto de signos salen y llegan
* Información: Conjunto de datos que se han colocado en un contexto significativo y útil.

Datos / información:
* Relevante. En cuanto a tiempo que se descubre-encuentra la información necesaria
* Precisa.
* Completa.
* Adecuada.
* Oportuna.
* Nivel de detalle adecuado
* Comprensible

Un sistema de info ejecuta tres act:
* Recibe datos de fuentes internas o externas de la empresa como entrada
* Actuá sobre los datos para producir info.
* Produce la info para el futuro usuario.

Elementos de un sistema:

* Componentes
* Las relaciones entre ellos, que determinan la estructura del sist.
* El objetivo
* El entorno del sistema.
* Los limites del sistema: frontera entre lo que es el sistema y lo que constituye el entorno

Características:

* Totalidad: su funcionamiento requiere unión de todos los subsistemas
* Búsqueda de Objetivos: para la permanencia del sistema este busca definir un sentido de unidad
* Equifinalidad: Es la propiedad de conseguir por caminos distintos los objetivos del sistema. El sistema tiene mas de una forma de lograr los objetivos
* Interrelación e Interdependencia: todos los elementos del sistema interactúan entre si y el resultado de cada uno de ellos depende por lo menos la actividad de sus elementos
* Jerarquía: Todo sistema contiene elementos los cuales a su vez cuentan con subelementos y todo el sistema a su vez es parte de un sistema mayor.
* Adaptabilidad: Es la capacidad del sistema para adaptarse a su entorno
* Eficiencia: Son los esfuerzos para utilizar los recursos en la mejor forma posible
* Sinergia: interacción de las partes individuales, se vuelve mas eficiente si cada parta actuá de manera aislada.

Funciones de un sistema de información dentro de un Sistema

Tres objetivos claros:
1. Automatización de procesos operativos
2. Proporcionar info que sirva de apoyo al proceso de la toma de decisiones.
3. Lograr ventajas competitivas a través de su implementación y uso.

Como resolver un proceso genérico?:
* Decidir que hacer
* Decidir como harcelo
* Hacerlo
* Probar el resultado
* Usar el producto

Sistema de Software

Pressman: el establecimiento y uso de principios de ingeniería robustos, orientados a obtener software económico que sea fiable y funcione de manera eficiente sobre maquinas reales

IEEE: el enfoque sistemático para el desarrollo, operación, mantenimiento y documentación del software

Definición entre definiciones: el conjunto de métodos, técnicas y herramientas que controlan el proceso integral del desarrollo del soft y suministra las bases para construir soft de calidad de forma eficiente en los plazos adecuados.

La ingeniería abarca:

* Métodos o técnicas. Es un proceso formal para producir algún resultado. Indican como construir técnicamente el soft y abarca una serie de tareas que incluyen la planificación y estimación de proyectos, análisis de requisitos, diseño, codigo, test y mantenimiento.

* Herramientas: puede ser instrumentos o sist automatizados para realizar algo de la mejor manera posible.

* Procedimientos: La combinación entre técnicas y las herramientas que en forma conjunta dan un resultado. Los procedimientos indicaran que herramientas deberán utilizar cuando se aplica cierta técnica.

* Paradigmas: Es la filosofía de la construcción del soft. Es el estilo de cocina.

Estudio de Factibilidad

Estructura de análisis de requisitos:

1, Actividades iniciales: análisis de necesidades, estudio de viabilidad

2, Técnicas para obtener información

3, Actividades generales de análisis

4, Documentos de especificación de requisitos(especificación de requisitos y casos de uso)

Qué es un proyecto; Un **proyecto** es esencialmente un conjunto de **actividades interrelacionadas**, con un inicio y una finalización definida, que utiliza **recursos** limitados para lograr un **objetivo** deseado.

Estudio de Factibilidad:

Versión comprimida del proceso total de análisis y diseño del sistema, es una herramienta que se utiliza para guiar la toma de decisiones en la evaluación de un proyecto. Identifica la probabilidad de éxitos o fracasos del soft.

El estudio comienza clarificando la definición del problema. Se confirma o se corrige la definición inicial de alcances y objetivos, y se identifica cualquier restricción impuesta sobre el sistema. Se pueden realizar estudios de mercado, de fondos e inversiones, etc.

Una vez generada la definición aceptable del problema, el analista desarrolla un modelo lógico del sistema y se buscan también soluciones alternativas para verificar la factibilidad.

Como se evalúan las alternativas?:

* Económica: Los beneficios superan los costos?
* Técnico: Puede implementarse el sistema usando la tecnología actual?
* Operativo: Puede implementarse el sistema en esta organización?

Los resultados del estudio de factibilidad se presentan al gerente y al usuario (informe escrito y son comunes las presentaciones orales). Una posibilidad es “abandone el proyecto”

El estudio de factibilidad debe proporcionar un sentido de dirección técnica para el proyecto, un plan de avanzar.

El costo del estudio sera aproximadamente de un 5 a un 10% del costo total del proyecto.

Los pasos de un estudio de factibilidad

1) Definir alcances y objetivos del sistema. Durante la definición del problema se preparó una definición de alcance y objetivos Confirmar la definición del problema. Cualquier restricción debe identificarse claramente. Buscar la respuesta a una pregunta simple: ¿Estoy trabajando en el problema correcto?
2) Estudio del sistema existente:
    - Analizar los procedimientos y la documentación
    - Seguir el flujo de trabajo, un buen punto de comienzo es la lista de distribución de los informes generados. Aprender que hace el sistema y por qué.
    - Entrevistar usuarios, aprender que el usuario describe síntomas y no problemas reales.

- No gastar mucho tiempo en analizar el sistema existente. El objetivo no es documentar lo que se hace sino entender lo que se hace.

3) Desarrolle un modelo lógico de alto nivel: El analista debe tener un buen sentido de las funciones y restricciones del sistema nuevo. Se puede construir un diagrama de flujo de datos y tal vez un diccionario de datos. Inclusive se puede usar más adelante este modelo lógico como diseño del sistema.

4) Redefinir el problema a la luz de los nuevos conocimientos:

- Desarrollando un modelo lógico del sistema propuesto. “esto es lo que yo pienso que debería de hacer el sistema”.
- El analista debe revisar la definición del problema, el alcance y los objetivos con el personal clave.
- Si el analista no ha comprendido, o el usuario ha pasado por alto algo, ahora es el momento de descubrirlo
- Piense en los primeros 4 pasos como un todo

5) Desarrollar y evaluar soluciones alternativas: Lo que se denomina factibilidad técnica.

6) Decidir sobre un curso de acción a recomendar. La decisión clave que surge del estudio de factibilidad es si se continua o se abandona.

7) Esbozar un plan de desarrollo, Asumiendo que la gerencia aceptó el curso de acción recomendado, se debe desarrollar un plan de implementación. Estimar requerimientos de personal, indicar la necesidad de analistas, programadores y otro personal técnico. Estimar costos de etapas en el ciclo de vida del sistema.

8) Redactar el estudio de factibilidad.

9) Presentar los resultados a la gerencia y usuario.

Análisis de Requisitos

“ Proceso de estudio de las necesidades de los usuarios para llegar a una definición de los requisitos del sistema, de hardware o software.”

“El proceso de estudio y refinamiento de dichos requisitos”

Según IEEE un requerimiento es la condición o capacidad que debe poseer un sistema o un componente de un sistema para satisfacer un contrato, un estándar, especificación u otro documento formalmente impuesto. La gestión de requerimientos comprende al conjunto de actividades que intenta entender las necesidades de los usuarios y traducirlas en afirmaciones precisas que se usarán en el desarrollo.

Los problemas con los requisitos constituyen la principal fuente de problemas 37%. Este proceso es la piedra angular en la construcción de software . El costo de solucionar un error en la etapa de mantenimiento es aprox 200 veces mayor que solucionarlo en la etapa de requerimientos.

Actividades generales del análisis de requisitos

Extracción o licitación de requisitos (técnicas para obtener información):

3) Análisis de requisitos
4) Especificación de requisitos
5) Validación de los requisitos (por parte de usuarios, comprobando la validez, consistencia y lo completo que sean)

El proceso de gestión de requerimientos implica tres tipos de tareas:

- Elicitación: Se trabaja estrechamente con los usuarios a fin de conocer la problemática en detalle. La esencia de esta etapa consiste en extraer el conocimiento relevante del problema
- Especificación: Es el proceso de documentación del comportamiento deseado del sistema. Una especificación puede ser vista como un acuerdo entre usuarios y desarrolladores.
- Validación: Permite asegurar que las especificaciones reflejan correctamente las intenciones del cliente y usuarios.

¿Quienes hacen ingeniería de Requerimientos?

Muy difícil encontrar a una persona que sepa entrevistar, escuchar, cuestionar, modelar, analizar, facilitar discusiones y negociaciones, observar, comunicar de manera verbal y escrita, relacionarse con gente, innovar etc.

Que tenga experiencia en el dominio del problema y de la solución.

Requisitos Funcionales y No Funcionales

Requisitos Funcionales: Describen la funcionalidad o los servicios que se espera que el sistema proveerá, sus entradas y salidas, excepciones, etc.

El requerimiento o requisito funcional describe que debe hacer el sistema respecto a su entorno. En otras palabras, refleja las necesidades de los usuarios o la interacción con otros sistemas.

Requisitos No funcionales: se refieren a las propiedades emergentes del sistema como la fiabilidad, el tiempo de respuesta, la capacidad de almacenamiento, la capacidad

de dispositivos de entrada/salida y la representación de datos que se utilizan en las interfaces del sistema.

Clasificación de Requerimientos

Clasificación modelo FURPS. Definición de Larman en 2002.

F: Functionality
U: Usability
R: Reliability
P: Performance
S: Supportability

Requerimientos de Funcionabilidad (Functionality): Son aquellos requerimientos que reflejan las características fundamentales (requerimiento funcional o funcionalidades del sistema), además de capacidades y seguridad.

Requerimientos de usabilidad o Capacidad de uso (Usability): son aquellos requerimientos que representan facilidad o nivel de uso del producto. Es decir, el grado en el que el diseño de un elemento facilita o dificulta su manejo. Se incluyen factores humanos, estética, consistencia de la interfaz, ayuda en linea, agentes y wizards, documentación y material de entrenamiento. Por ej, visibilidad del texto a una cierta distancia y combinación de colores del texto

Requerimientos de Fiabilidad (Reliability): son aquellos que muestran la capacidad de un sistema o componente para ejecutar las funciones requeridas bajo condiciones normales en un periodo de tiempo especifico. Teniendo en cuenta sub-categorías como: Disponibilidad(porcentaje de tiempo disponible, horas de uso, etc), tiempo medio entre fallas, tiempo medio para reparación, cuanto tiempo puede es posible que el sistema esté inoperante después que falla, exactitud (precisión y exactitud, según algún estándar conocido) que se requiere para las salidas del sistema, cantidad máxima de errores o porcentaje de defectos, etc.

Requerimientos de Desempeño(Performance): se refieren a las características de rendimiento del sistema. Incluye tiempos de respuesta específicos. Por ej: Tiempo de respuesta para una transacción, transacciones por segundo, capacidad, como opera el sistema si es degradado de alguna manera, como utiliza los recursos, etc.

Requerimientos de Capacidad de Soporte(Supportability): son requerimientos que refuerzan el soporte y mantenimiento del sistema que está siendo construido, incluyendo normas de codificación, convenciones de nombres, librerías, acceso para mantenimiento, utilidades de mantenimiento si las hay. Como requerimiento que ayuda al mantenimiento se debe hacer referencia al uso de nomenclatura común para el desarrollo del sistema y a la metodología de desarrollo.

Como debe ser un requerimiento

Un requerimiento debe ser:

- Especificado por escrito, como todo contrato o acuerdo entre dos partes.
- Posible de probar o verificar, si un requerimiento no se puede comprobar entonces, como se sabe si se cumplió con él o no?
- Conciso, un requerimiento es conciso si es fácil de leer y entender. Su redacción debe ser simple y clara para aquellos que vayan a consultarlo en el futuro.

Documentos de especificación de requisitos

Después de realizar el informe de necesidades y dar luz verde al proyecto, se crea el SyRS (System Requirements Specification) y el SRS(Software Requerements Specification)

Se espera que una especificación de requerimientos que fue aprobada por clientes y/o usuarios tenga al menos las siguientes características:

- que contenga todos los requerimientos deseados
- que cada requerimiento solo tenga una interpretación posible
- que el cumplimiento de cualquier requerimiento no provoque conflictos con el cumplimiento de otro requerimiento, es decir, que sea consistente
- que se definan prioridades

Como formalizar la especificación de requerimientos

1. Definición de misión y alcance: Si no se sabe a dónde se está yendo, será muy difícil que finalmente se llegue al lugar correcto. Se sugiere comenzar redactando un pequeño párrafo que sintetice claramente qué se espera del proyecto y que especifique qué se espera finalmente del software a desarrollar. Detallando el alcance general deseado para el mismo. El estilo de redacción tiene que estar orientado para que sea fácilmente entendido por las autoridades o responsable de áreas que están solicitando el software nuevo o la ampliación de uno existente. También es conveniente definir un nombre para el proyecto o software a desarrollar
2. Determinar quiénes son los actores relevantes: El software implementa funcionalidades que fueron solicitadas por alguien. Identificar todos los posibles beneficiarios / usuarios del soft, agrupados por perfiles, permite minimizar el riesgo de que el soft esté incompleto en términos de funcionalidades requeridas.
   Resulta necesario distinguir y jerarquizar los beneficiarios de manera de concentrar los esfuerzos sobre las personas más relevantes.

3. Determinar Objetivos: El refinamiento de los objetivos es un proceso incremental. Ya que es habitual que las primeras definiciones sean de un carácter genérico y hasta de cierta ambigüedad. Podemos entonces definir objetivos generales y avanzar paulatinamente en la definición de objetivos particulares que resulente más concretos y verificables en términos de su realización. Avanzando desde lo general a lo particular, y específicamente a lo que pretenda lograr cada usuario/beneficiario del sistema, podremos encontrar objetivos que resulten conflictivos entre sí. Cuanto más rápido se puedan identificar estas situaciones más sencilla será su solución.
4. Descripción del circuito funcional: El soft a desarrollar siempre está inserto en un determinado circuito administrativo que incluye personas, procedimientos, comprobantes, otros sistemas, etc. Describir correctamente este circuito administrativo no sólo le sirve al analista funcional que está trabajando en determinar los requerimientos, sino que también sirve como marco de referencia para el resto de las personas que integran el equipo de desarrollo, que no necesariamente participan del relevamiento funcional, no tienen contacto con los usuarios ni conocen el contexto en que deberá funcionar el soft a desarrollar.
5. Detallar principales prestaciones funcionales: Se sugiere detallar una lista de prestaciones funcionales que el software debe cumplir, agrupadas por perfiles de usuarios (“actores”). Cada prestación funcional debe estar definida en uno o dos renglones.
6. Especificar requerimientos: Describir el comportamiento esperado para cada una de las prestaciones funcionales detalladas en el punto anterior. Se recomienda redactarlo desde el punto de vista del usuario, poniendo foco en los aspectos conceptuales de lo que debe hacer y no tanto en detalles sobre comportamiento interno del soft, ya que puede resultar de interés para el equipo de desarrollo pero no tanto para el usuario final. Cuanto más conceptual sea la descripción mas fácil será de mantener actualizado, y podría ser un buen insumo para redactar el manual conceptual del sistema. Se sugiere utilizar el formato de “caso de usos” y complementarlo con un diagrama de interacción cuando se lo considere oportuno.
7. Determinar requerimientos no funcionales: Es necesario prestar especial atención a aquellas cosas que deben ser tenidas en cuenta a la hora del diseño del sistema, más allá de la especificación funcional detallada que se realice. En general se dice que los requerimientos no funcionales son “restricciones”
   EJ: “el desarrollo debe ser en PHP y PSQL”
   Es importante que este tipo de requerimientos no contengan ambigüedades. EJ : “Se desea que el sistema sea rápido y que presente una interfaz amigable para el usuario”.
8. Determinar prioridades: Como definir la secuencia en que deben desarrollarse las prestaciones funcionales. Una opción es que el equipo de desarrollo lo defina utilizando su propio criterio, lo cual no es recomendable

porque dificilmente sea concordante con las necesidades reales. Se recomienda que las prioridades queden formalmente definidas por los propios interesados. De esta manera se pueden planificar prototipos o ciclos de desarrollo incrementales que respondan a las necesidades operativas.

Ciclo de vida de Software

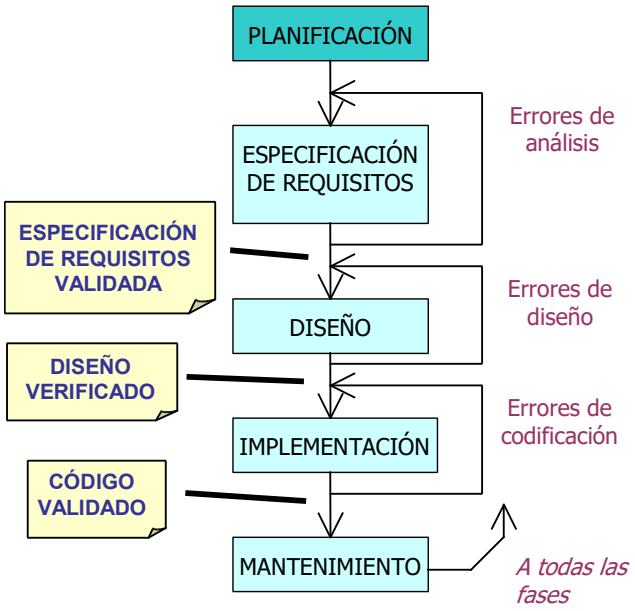
Concepto de ciclo de vida:

- Es una descripción de un proceso de software que se presenta desde una perspectiva particular.
- Es una abstracción de un proceso real
- Existe una gran variedad de modelos diferentes “genéricos” o paradigmas de desarrollo de software

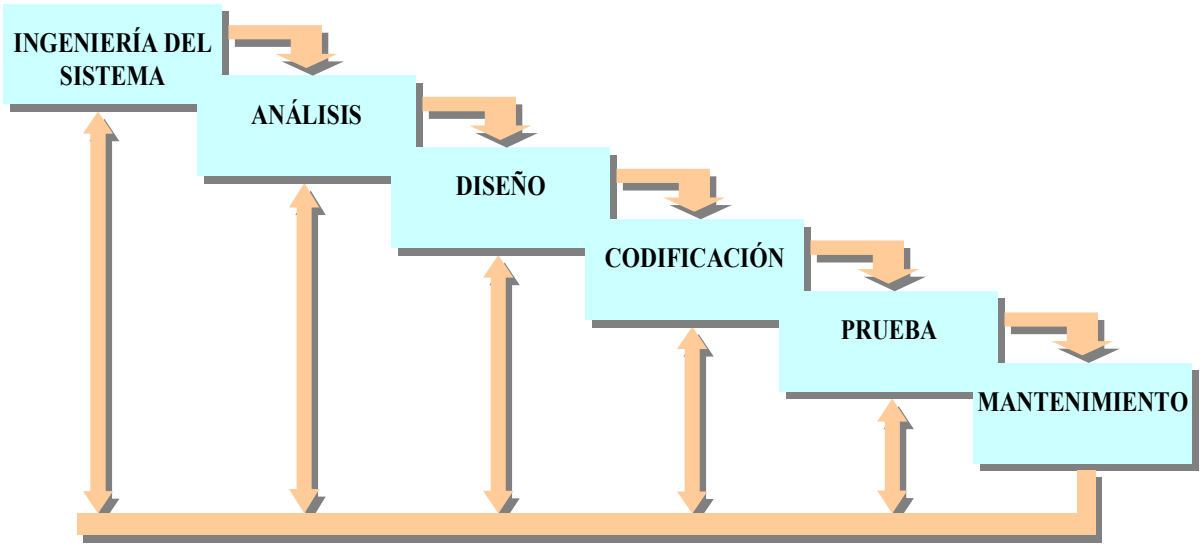
Modelo en Casada:

- Primer modelo empleado (Royce 1970)
- También denominado: “ciclo de vida clásico”, “paradigma clásico”, “orientado a fases”, “lineal secuencial”
- Ejecución secuencial de una serie de fases
- Cada fase genera documentación para la siguiente
- Varias propuestas

modelo en casada ideal:



definición alternativa:



Es similar al enfoque de ingeniería según el cual se construyen los edificios o los puentes (Larman 2003). → Sin embargo no siempre funciona en el software

Fases del modelo en cascada

- Análisis del sistema:

  * ¿Qué debe hacer el sistema?
  * El soft suele formar parte de un sistema mayor. Se deben identificar los requisitos de todos los elementos del sistema, diseñar la arquitectura del sistema, asignar un subconjunto de dichos requisitos al soft

- Análisis de los requisitos del software:
  * El proceso de recopilación de los requisitos se centra especialmente en el soft
  * Hay que especificar:
  - Funciones que el soft debe realizar
  - la información que el soft va a gestionar
  - Condiciones existentes: rendimiento, utilización de recursos, etc
  * Los requisitos del soft se documentan y se revisan con el cliente. Generando así ERS (especificación de requisitos del Soft)

- Diseño:
  * Cómo se ha de construir el sistema?
  * Definir la estructura del software que satisfaga los requisitos con la calidad necesaria:

- Estructura de datos
- Arquitectura del software
- Representaciones de interfaz
- Determinar los algoritmos
* Diseño preliminar (arquitectónico)
* Diseño detallado

- Generación de código: A veces se puede realizar de forma automática a partir de un diseño detallado

- Test: Se ha construido el sistema que se deseaba?
  * Prueba interna o de caja blanca / negra
  * Test unitarios, de integración, del soft, del sistema, de aceptación

- Mantenimiento:
  * El soft evolucionará, sufrirá cambios después de que se entregue al cliente (errores, nuevas funciones, aumentos del rendimiento, etc)
  * Volver a aplicar cada una de las fases anteriores al programa existente

Puntos a favor del modelo en cascada

- Tener en cuenta que fue el primer modelo empleado " es mejor que nada"
- Proporciona un marco para aplicar métodos, técnicas y herramientas
- Es "sencillo" controlar qué productos se deben generar durante el desarrollo del proyecto
- Para un proyecto corto (ej dos meses) se puede usar un ciclo de vida en cascada (Larman 2003) → Las dificultades aparecen cuando la escala de tiempo se alarga

Críticas

Cuando el proyecto se alarga:

9. La complejidad pasa a ser alta
10. Las decisiones especulativas se incrementan y complican, ya que los requisitos se congelan y no existe retralimentación a partir de implementaciones y pruebas reales
11. En general, las cuestiones de alto riesgo no se abordan lo suficientemente pronto, no existe un intento activo de identificar y mitigar en primer lugar las cuestiones de más riesgo

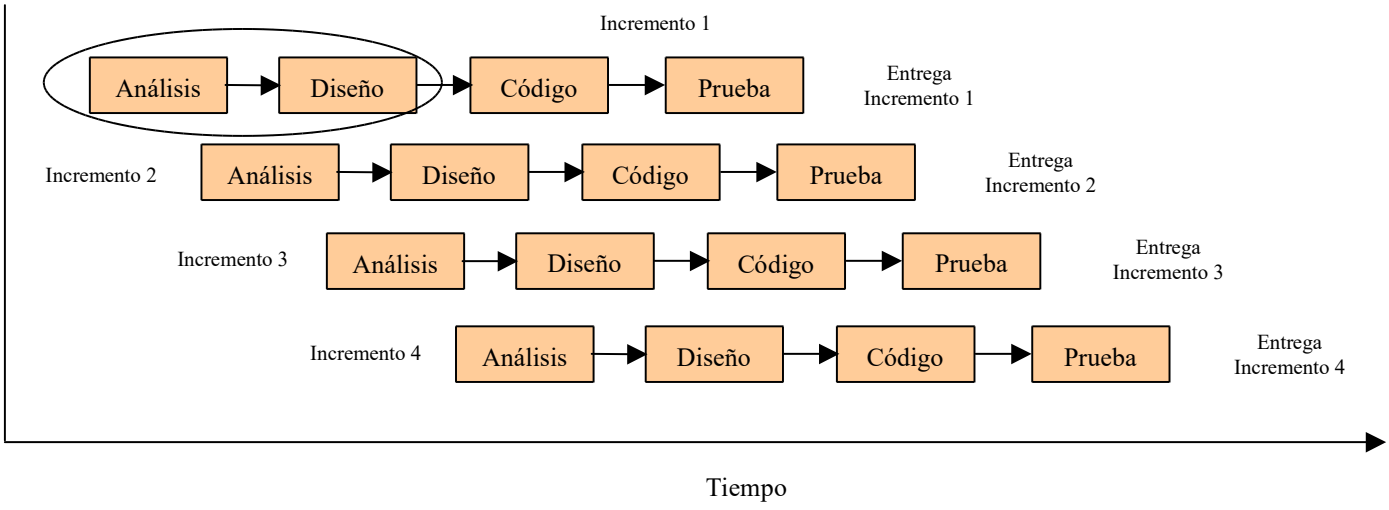
Otras consideraciones:

- Es poco realista, los proyectos reales raramente pueden seguir el flujo secuencial que se propone
- En general, establecer todos los requisitos al principio del proceso es un “mito inalcanzable”
  - Los requisitos no se pueden “congelar”, la única constante es el cambio!
  - “Lo sabré cuando lo vea”: las personas involucradas cambian de idea o no pueden imaginar lo que quieren hasta que no ven un sistema concreto
  - El mercado cambia

- La atención en ingeniería del software está en modelos de proceso iterativos e incrementales, que ayudan a mitigar los problemas del paradigma clásico (NO SILVER BULLET), las estructuras conceptuales que construimos hoy son demasiado complicadas para que se especifiquen con precisión por adelantado y muy complejas para desarrollar sin errores. Debemos ir en un modelo en espiral
- “ el usuario debe tener paciencia”
- Se tarda mucho tiempo en pasar por todo el ciclo, hasta que no termina una fase no empieza la siguiente
- Los errores de análisis y diseño son difíciles de eliminar y se propagan a las etapas siguientes con un efecto conocido como “bola de nieve”
- En la práctica, el modelo tiende a deformarse y todo el peso de la validación y mantenimiento recae, en su mayor parte, sobre el código fuente
- El soft va deteriorándose y resulta cada vez más difícil de mantener

Modelo Incremental

Pressman 2002 pp. 23-24



El **modelo incremental** de gestión de proyectos tiene como objetivo un crecimiento progresivo de la funcionalidad. Es decir, el producto va evolucionando con cada una de las entregas previstas hasta que se amolda a lo requerido por el cliente o destinatario.

En cada una de ellas, **el producto debe mostrar una evolución con respecto a la fecha anterior;** nunca puede ser igual.

Una de las claves para que esto se haga efectivo es la evaluación de las etapas. Los responsables del proyecto deben analizar si los resultados parciales son los esperados y si, sobre todo, apuntan al objetivo principal. De no ser así, deberán intervenir en él e implementar las soluciones que la situación requiera.

Características

- Los incrementos son pequeños
- Permite una fácil administración de las tareas en cada iteración.
- La inversión se materializa a corto plazo.
- Es un modelo propicio a cambios o modificaciones.
- Se adapta a las necesidades que surjan.

Pasos del modelo

1. Requerimientos: son los objetivos centrales y específicos que persigue el proyecto.
2. Definición de las tareas y las iteraciones: teniendo en cuenta lo que se busca, el siguiente paso es hacer una lista de tareas y agruparlas en las iteraciones que tendrá el proyecto. Esta agrupación no puede ser aleatoria. Cada una debe perseguir objetivos específicos que la definan como tal.
3. Diseño de los incrementos: establecidas las iteraciones, es preciso definir cuál será la evolución del producto en cada una de ellas. Cada iteración debe superar a la que le ha precedido. Esto es lo que se denomina incremento.
4. Desarrollo del incremento: posteriormente se realizan las tareas previstas y se desarrollan los incrementos establecidos en la etapa anterior.
5. Validación de incrementos: al término de cada iteración, los responsables de la gestión del proyecto deben dar por buenos los incrementos que cada una de ellas ha arrojado. Si no son los esperados o si ha habido algún retroceso, es necesario volver la vista atrás y buscar las causas de ello.

6. Integración de incrementos: una vez son validados, los incrementos dan forma a lo que se denomina línea incremental o evolución del proyecto en su conjunto. Cada incremento ha contribuido al resultado final.
7. Entrega del producto: cuando el producto en su conjunto ha sido validado y se confirma su correspondencia con los objetivos iniciales, se procede a su entrega final.

Modelo en Espiral

Boehm 1986

Las actividades de este modelo se conforman en una espiral, en la que cada bucle o iteración de 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.

El modelo Espiral tiene en cuenta fuertemente el **riesgo** que aparece a la hora de desarrollar software, analizando las opciones mas viables en cuanto a riesgo y optando por la menor.

Por cada ejecución de una iteración existen 4 pasos:

1. Determinar o fijar los objetivos. En este paso 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.
2. Análisis del riesgo. En este paso se es poco realista, los proyectos reales rara vez pueden seguir el flujo secuencial
3. 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.
4. 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
5. 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.

Con cada iteración alrededor de la espiral, se crean sucesivas versiones del software, cada vez más completas y, al final, el sistema de software ya queda totalmente funcional. La diferencia principal entre el modelo espiral y los modelos anteriores (ej.: cascada, evolutivo, incremental, etc.) es la evaluación del riesgo. El riesgo es todo lo que pueda salir mal en un proyecto de desarrollo de software. Por ejemplo, si queremos utilizar un lenguaje de programación para desarrollar un sistema operativo, un riesgo posible es que los compiladores utilizables no produzcan un código objeto eficiente. Los riesgos originan problemas en el proyecto, como el exceso de los costos. Es así que, la disminución de los riesgos es una actividad muy importante.

Un modelo espiral comienza con la determinación de los objetivos tanto funcionales como de rendimiento. Después se enumeran algunas formas posibles de alcanzar estos objetivos identificando las fuentes de riesgos posibles. Luego continuamos con el siguiente paso que es resolver estos riesgos y llevar a cabo las actividades de desarrollo, para finalizar con la planificación del siguiente ciclo de la espiral.

Características

Es considerado como un modelo evolutivo ya que combina el modelo clásico con el diseño de prototipos.

- Contiene una nueva etapa que es el análisis de riesgos, no incluida anteriormente
- Este modelo es el indicado para desarrollar software con diferentes versiones actualizadas como se hace con los programas modernos de PC´s
- La ingeniería puede desarrollarse a través del ciclo de vida clásico o el de construcción de prototipos
- Este es el enfoque más realista actualmente

Ventajas:

- No requiere una definición completa de los requerimientos del software a desarrollar para comenzar su funcionalidad
- En la terminación de un producto desde el final de la primera iteración es muy factible aprobar los requisitos
- Sufrir retrasos corre un riesgo menor, por que se comprueban los conflictos presentados tempranamente y existe la forma de poder corregirlos a tiempo.

Desventajas:

- Existe complicación cuando se evalúa los riesgos.
- Se requiere la participación continua por parte del cliente.
- Se pierde tiempo al volver producir inicialmente una especificación completa de los requerimientos cuando se modifica o mejora el software.

Proceso Unificado

(Ivar Jacobson, Grady Booch y James Rumbaugh)

El Proceso Unificado es un proceso de desarrollo de software: "conjunto de actividades necesarias para transformar los requisitos del usuario en un sistema software"

Es una metodología de desarrollo de software que está basado en componentes e interfaces bien definidas, y junto con el Lenguaje Unificado de Modelado (UML),

UP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización.

Dirigido por Casos de Uso: Basándose en los casos de uso los desarrolladores crean una serie de modelos de diseño e implementación que llevan a cabo los casos de uso. De este modo los casos de uso no solo inician el proceso de desarrollo sino que le proporcionan un hilo conductor

Principales elementos dentro de UP:

- Quien?: Define el comportamiento y responsabilidades (rol) de un individuo, grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto como un equipo
- Cómo?: Es una tarea que tiene un propósito claro, es realizada por un trabajador y manipula elementos.
- Qué?: Productos tangibles del proyecto que son producidos, modificados y usados por las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables.
- Cuándo?: Secuencia de actividades realizadas por trabajadores y que produce un resultado de valor observable.

Características:

- Unifica los mejores elementos de metodologías anteriores.
- Preparado para desarrollar grandes y complejos proyectos.
- Orientado a Objetos.
- Utiliza el UML como lenguaje de representación visual.

Ventajas:

- Coste del riesgo a un solo incremento.
- Reduce el riesgo de no sacar el producto en el calendario previsto.
- Acelera el ritmo de desarrollo.
- Se adapta mejor a las necesidades del cliente.

Estructura del ciclo de vida:

- Dirigido por los casos de uso: Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos. A partir de aquí los casos de uso guían el proceso de desarrollo ya que los modelos que se obtienen
- Centrado en la arquitectura: La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo. Se desarrolla mediante iteraciones, comenzando por los CU relevantes desde el punto de

vista de la arquitectura. El modelo de arquitectura se representa a través de vistas en las que se incluyen los diagramas de UML.

- Iterativo e incremental: Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros.Las iteraciones deben estar controladas. Esto significa que deben seleccionarse y ejecutarse de una forma planificada.

Fases del desarrollo en UP: El desarrollo en UP es a través de fases, Cada fase se subdivide en iteraciones. En cada iteración se desarrolla en secuencia un conjunto de disciplinas o flujos de trabajos. Cada fase finaliza con un hito (entrega de sistema, cumplir determinado objetivo, etc)

Fase de Inicio: tiene por finalidad definir la visión, los objetivos y el alcance del proyecto, tanto desde el punto de vista funcional como del técnico, obteniéndose como uno de los principales resultados una lista de los casos de uso y una lista de los factores de riesgo del proyecto. El objetivo de esta fase es ayudar al equipo de proyecto a decidir cuales son los verdaderos objetivos del proyecto. Por resultado se pueden obtener: mayores requerimientos, boceto inicial, objetivos generales, versión muy preliminar, modelo de negocio, etc.

Fase de elaboración: Tiene como principal finalidad completar el análisis de los casos de uso y definir la arquitectura del sistema, además se obtiene una aplicación ejecutable que responde a los casos de uso que la comprometen.

La fase de elaboración finaliza con el hito de la Arquitectura del Ciclo de Vida. Este hito se alcanza cuando el equipo de desarrollo llegan a un acuerdo como: casos de uso que describen la funcionalidad del sistema, la línea base de arquitectura, los riesgos, el plan del proyectos

Fase de Construcción: está compuesta por un ciclo de varias iteraciones, en las cuales se van incorporando sucesivamente los casos de uso, de acuerdo a los factores de riesgo del proyecto. En la construcción se crea el producto. La línea base de la arquitectura crece hasta convertirse en el sistema completo. Al final de esta fase, el producto contiene todos los casos de uso implementados, sin embargo puede que no este libre de defectos.

Al final de la fase se obtiene: el soft, casos de test, manuales de usuarios

Fase de transición: Se inicia con una versión “beta” del sistema y culmina con el sistema en fase de producción. Las iteraciones en esta fase continúan agregando características al sw. Sin embargo las características se agregan a un sistema que el usuario se encuentra utilizando activamente.

La fase de transición finaliza con el hito de Lanzamiento del Producto.

1) **Que se evalúa en un estudio de factibilidad de un proyecto de software.**

   Económica: *los beneficios ¿superan los costos?*
   Técnico: *¿Puede implementarse el sistema usando la tecnología actual?*
   Operativo: *¿Puede implementarse el sistema en esta organización?*

2) **Que es un sistema de información.**
   Un Sistema de Información es un conjunto de datos que interactúan entre sí con un fin común.

3) **Por qué se deben documentar los requisitos y como se realiza.**
   Porque solucionar un problema o error en la etapa de requerimientos es 200 veces menos costoso que en la etapa de mantenimiento.

   Analisis de requisitos consta de 4 etapas:
   1- Actividades iniciales:
      a. Analisis de necesdades.
      b. Estudo de viabilidad.
   2- Técnicas para obtener información.
   3- Actividades generales de análisis.
   4- Documento de especificación de requisitos:
      a. Especificacion de requisitos.
      b. Casos de uso.

4) **¿Qué es la especificación de requisitos de software?**

5) **¿Cuáles son las 4 etapas indispensables en la ingeniera de requisitos? Desarrolle.**
   1- Actividades iniciales:
      a. Analisis de necesdades.
      b. Estudo de viabilidad.
   2- Técnicas para obtener información.
   3- Actividades generales de análisis.
   4- Documento de especificación de requisitos:
      a. Especificacion de requisitos.
      b. Casos de uso.

6) ¿Qué características tienen los sistemas de información transaccionales?

Transaccionales:
Mas importantes, los que generan más trabajo.
Transacción: suceso que afecta a toda la organización. Facturación, pagos, depósitos, etc.
Sistema de Información para la Gestión
Suelen introducirse después de haber implantado los sistemas transaccionales.
Suelen ser intensivos en cálculos y escasos en entradas y salidas de información.
Ayudan a los directivos a tomar decisiones y resolver problemas
Sistemas de Apoyo a las Decisiones
Su función principal no es apoyar a la automatización de procesos operativos ni proporcionar información para la toma de decisiones. Sin embargo, este tipo de sistemas puede llevar a cabo dichas funciones

7) ¿Cuáles son las diferencias entra entrevista, muestreo y JAD?

Entrevistas:
Es una conversación entre dos o más personas dirigida a satisfacer las necesidades de información del entrevistador.

- Esta técnica consta de reunir sólo un conjunto representativo de los datos.
- Es apropiada cuando hay un gran volumen de datos.

JAD:
Conjunto de reuniones usuarios/analistas:
2 - 4 días
Dinámica de grupos
Se comienza con un documento de trabajo, y se discute ➔ Al dinal del JAD ➔ Doc. de Requisitos(aprobado).

8) ¿Cuáles son los pasos del estudio de la factibilidad?

- Definir alcances y objetivos.
- Estudio del sistema existente.
- Desarrollar un modelo lógico del sistema actual.
- Redefinir el Problema.
- Desarrollar y evaluar soluciones alternativas.
- Decidir sobre un curso de acción a recomendar.
- Esbozar un plan de desarrollo
- Redactar el estudio de factibilidad.
- Presentar los resultados la gerencia y usuario.

9) **¿En qué ciclo de vida se contempla el análisis de riesgo de algo fundamental? Que característica tiene este ciclo.**

En el modelo en Espiral.
Características: Es considerado como un modelo evolutivo ya que combina el modelo clásico con el diseño de prototipos.

- Contiene una nueva etapa que es el análisis de riesgos, no incluida anteriormente
- Este modelo es el indicado para desarrollar software con diferentes versiones actualizadas como se hace con los programas modernos de PC´s
- La ingeniería puede desarrollarse a través del ciclo de vida clásico o el de construcción de prototipos
- Este es el enfoque más realista actualmente

10) **¿Cuáles son las fases del proceso unificado? Nombre las principales actividades por fase junto a su hito.**

Fase de Inicio: tiene por finalidad definir la visión, los objetivos y el alcance del proyecto, tanto desde el punto de vista funcional como del técnico, obteniéndose como uno de los principales resultados una lista de los casos de uso y una lista de los factores de riesgo del proyecto. El objetivo de esta fase es ayudar al equipo de proyecto a decidir cuáles son los verdaderos objetivos del proyecto. Por resultado se pueden obtener: mayores requerimientos, boceto inicial, objetivos generales, versión muy preliminar, modelo de negocio, etc.

Fase de elaboración: Tiene como principal finalidad completar el análisis de los casos de uso y definir la arquitectura del sistema, además se obtiene una aplicación ejecutable que responde a los casos de uso que la comprometen.

La fase de elaboración finaliza con el hito de la Arquitectura del Ciclo de Vida. Este hito se alcanza cuando los equipos de desarrollo llegan a un acuerdo como: casos de uso que describen la funcionalidad del sistema, la línea base de arquitectura, los riesgos, el plan del proyecto

Fase de Construcción: está compuesta por un ciclo de varias iteraciones, en las cuales se van incorporando sucesivamente los casos de uso, de acuerdo a los factores de riesgo del proyecto. En la construcción se crea el producto. La línea base de la arquitectura crece hasta convertirse en el sistema completo. Al final de esta fase, el producto contiene todos los casos de uso implementados, sin embargo, puede que no esté libre de defectos.

Al final de la fase se obtiene: el soft, casos de test, manuales de usuarios

Fase de transición: Se inicia con una versión “beta” del sistema y culmina con el sistema en fase de producción. Las iteraciones en esta fase continúan agregando características al sw. Sin embargo, las características se agregan a un sistema que el usuario se encuentra utilizando activamente.

La fase de transición finaliza con el hito de Lanzamiento del Producto.