UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN

AUTOR: RENÉ MONTESANO BRAND

DESARROLLO DE *SOFTWARE* EMPRESARIAL		Clave: 1864
Plan: 2005		Créditos:8
Licenciatura: Informática		Semestre:8
Área: Informática (Gestión de la información)		Horas de asesoría:
Requisitos: Ninguno		Horas por semana: 4
Tipo de asignatura:	Obligatoria ( )	Optativa (x)

TEMARIO OFICIAL (horas sugeridas: 64)

	Horas
1. Sistemas de información en las empresas	4
2. Análisis de sistemas	12
3. Diseño de sistemas	8
4. Desarrollo de sistemas	8
5. Implementación de sistemas	8
6. Pruebas	4
7. Manual de usuario	8
8. Caso práctico	12

INTRODUCCIÓN GENERAL A LA ASIGNATURA

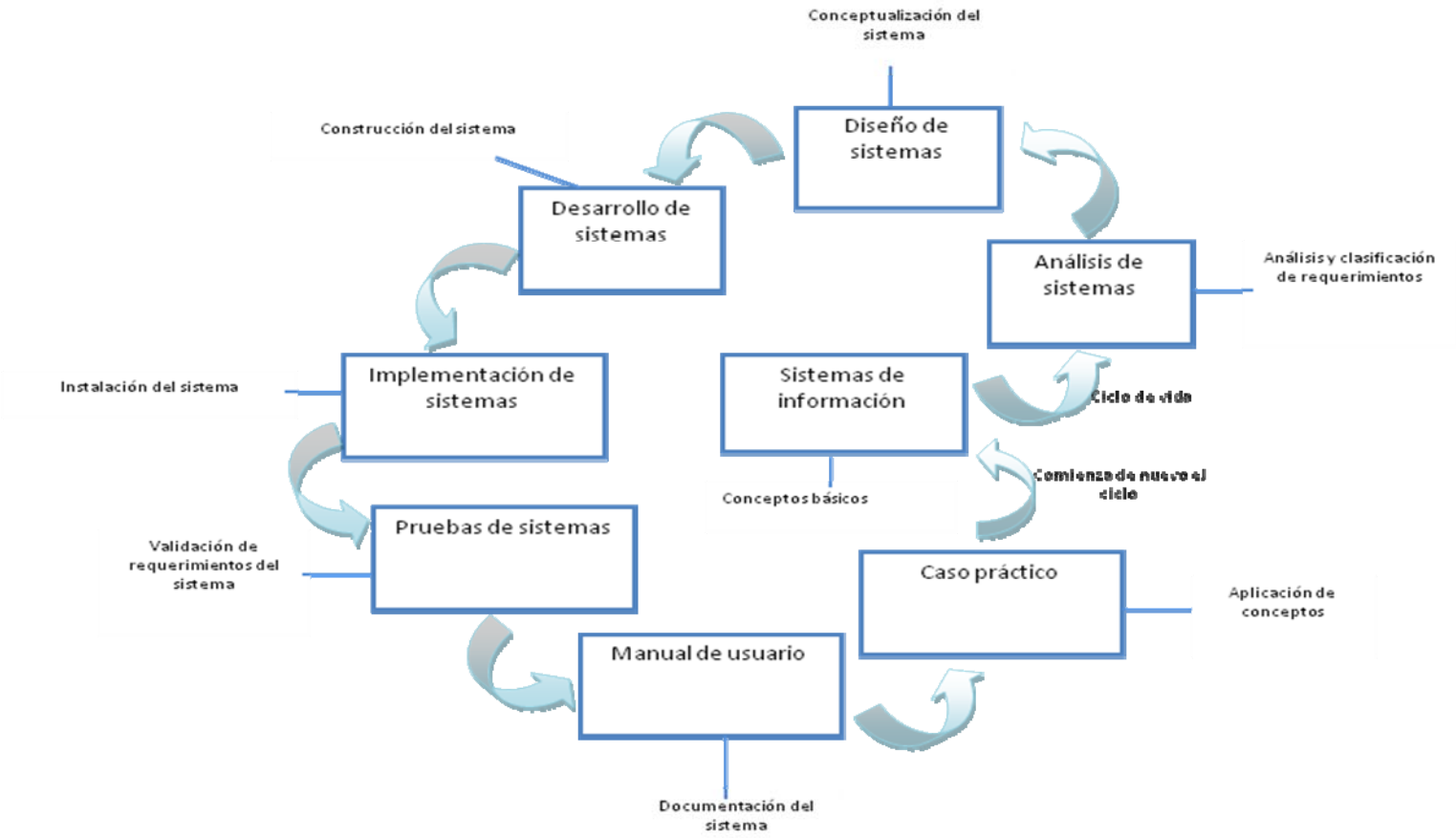
El mundo está rodeado de elementos intangibles que hacen más sencilla nuestra vida. Procesos que hasta hace poco tiempo eran considerados obligatorios, tediosos e implicaban mucho tiempo, se han simplificado al punto de sólo oprimir un botón desde un dispositivo tan común como un celular. ¿Qué hay detrás de todo esto? Una serie de componentes que implican creatividad, capital humano, tecnología y *software*, en particular empresarial. Éste se entiende como cualquier tipo de herramienta, aplicación, programa o desarrollo de *software* cuya finalidad sea mejorar la productividad, las comunicaciones, la relación con el cliente y la edición o generación de contenidos de una organización o empresa. Engloba asimismo una mezcla de programas informáticos que pueden abarcar aplicaciones comunes de negocios, herramientas de modelado, herramientas de desarrollo y rutinas específicas para la resolución de problemas. Y su impacto es medido según el tiempo y recursos ahorrados, así como su facilidad de uso respecto a la implantación de otras técnicas o herramientas, siempre respetando la lógica del negocio al que brinda soporte.

En este contexto, se analizan en este material los elementos necesarios para la generación y gestión de *software* en las empresas, así como las técnicas requeridas para realizar todo el proceso de resolución de problemas.

OBJETIVO GENERAL DE LA ASIGNATURA

Al finalizar el curso, el alumno desarrollará *software* siguiendo el ciclo clásico de sistemas: análisis, diseño, desarrollo, implementación y pruebas, con el objeto de resolver problemas de información en las empresas.

ESTRUCTURA CONCEPTUAL



UNIDAD 1. SISTEMAS DE INFORMACIÓN EN LAS EMPRESAS

OBJETIVO ESPECÍFICO

Al finalizar la unidad, el alumno reconocerá los elementos humanos y la tecnología que en conjunto son claves fundamentales para resolver las necesidades de las organizaciones empresariales.

INTRODUCCIÓN

El hombre siempre ha buscado formas o métodos para facilitar sus actividades y crea tecnologías que le ayudan a desempeñar diversas tareas o procesos. El campo empresarial no es excepción, las empresas buscan medios que hagan más productivos sus procesos internos y las lleven a establecer un dominio sobre sus mercados metas y un liderazgo importante en su rama. En este orden, el *software* empresarial es una tecnología que ha permitido a las empresas hacer más eficientes sus procesos internos, ayudando a reducir tiempos y costos, generando con ello mayores ganancias.

A lo largo de esta unidad, revisaremos la forma como han ido evolucionando los métodos para la generación de *software* empresarial, sus características, la complejidad que conlleva su desarrollo y, sobre todo, la importancia del factor humano para su diseño y construcción.

LO QUE SÉ

Desarrolla un mapa mental que muestre el ciclo de interacción entre usuarios, empresas y tecnología; y el *software* como vínculo entre ellas.

TEMARIO DETALLADO (4 horas)

1.1. Conceptos generales
1.2. Evolución histórica de la ingeniería de *software*
1.3. El *software* empresarial, su naturaleza y complejidad
1.4. Factores humanos que intervienen en la ingeniería del *software*
1.5. Modelos para el desarrollo de sistemas
1.6. Los sistemas de información como una ventaja competitiva en las organizaciones

1.1. Conceptos generales

Los sistemas de información empresarial (SIE) consisten en sistemas de información que, como su nombre indica, son enfocados a la administración del flujo de datos dentro de una organización. Su principal objetivo es ofrecer información con prontitud, confianza y asertividad a todas sus dependencias.

Asimismo, tener un alcance corporativo es considerado un elemento estratégico para dar soporte a los procesos de negocio. Este soporte sebe cumplir los siguientes objetivos:

- Dar valor como activos o recursos corporativos a los datos de la empresa.

- Ser la fuente primordial de información empresarial para todos sus usuarios y dependientes (las personas que requieran información para completar sus funciones dentro de la organización).

Su valor dentro del contexto empresarial está implícito en los datos que produce y auxilia en la toma de mejores decisiones en todos los niveles donde está presente. El procedimiento para manejar el flujo de información, su gestión, operación y almacenamiento, así como los datos generados por éste, hacen de los SIE un elemento central para toda organización que decide emplearlos.

1.2. Evolución histórica de la ingeniería de *software*

¿Qué es la ingeniería del *software*?

Se entiende por ingeniería de *software* el conjunto de métodos, herramientas, técnicas y procesos necesarios para la generación de programas (*software*) empleados en entornos informatizados. Como el *software* en gran medida es elaborado por el ser humano, Benet[1] aclara que el proceso de su creación no debe ser considerado una obra de arte, sino un producto de consumo masivo que otorga valor para una empresa o un trabajador autónomo.

Métodos de la ingeniería de *software*[2]

Para construir la ingeniería del *software* adecuadamente, se debe definir un proceso o método de desarrollo. Éste se puede dividir en tres fases genéricas, con independencia del área de la aplicación, tamaño o complejidad del proyecto: definición, desarrollo y mantenimiento.

En la definición, se intenta identificar qué información ha de ser procesada, qué función y rendimiento se desea, qué comportamiento del sistema, qué interfaces van a ser establecidas, qué restricciones de diseño existen y qué criterios de validación se necesitan para definir un sistema correcto. Así, deben puntualizarse los requisitos clave del sistema y del *software*. Aunque los métodos durante la fase de definición varían según los paradigmas de ingeniería de *software* que se apliquen, éstos darán lugar a tareas específicas: ingeniería de sistemas o de información, planificación del proyecto del *software* y análisis de los requisitos.

---

[1] Benet Campderrich Falgueras. *Ingeniería de* software. Disponible en http://goo.gl/Ikk4bs Recuperado: 08/10/2013.
[2] Pressman, R. *Ingeniería del* software. *Un enfoque práctico, 5.ª ed.* España: McGraw-Hill, 2002, p. 17.

La siguiente fase, desarrollo, se centra en el *cómo.* Aquí, el ingeniero del *software* intenta definir cómo ha de diseñarse las estructuras de datos, construirse la función como una arquitectura del *software*, implantarse detalles sobre los procedimientos, caracterizarse las interfaces, traducirse el diseño en el lenguaje de programación y realizarse las pruebas. Estos métodos durante la fase de desarrollo variarán, no obstante que las siguientes tareas ocurrirán siempre, en el diseño del *software*, la generación de código y la prueba del *software*.

Finalmente, la fase de mantenimiento se centra en el cambio, que va asociado a la corrección de errores, adaptaciones requeridas a medida que evoluciona el entorno del *software*, y a modificaciones por mejoras producidas por los requisitos cambiantes de las necesidades del cliente. En este momento, se vuelven a aplicar los pasos de las fases de definición y desarrollo, pero en el contexto del *software* éstas ya existen.

En la fase de mantenimiento, se dan cuatro tipos de cambios:

- *Corrección*. El mantenimiento correctivo modifica el *software* para enmendar los defectos.
- *Adaptación.* El mantenimiento adaptativo produce modificación en el *software* para acomodarlo a los cambios de su entorno externo.
- *Mejora.* Se lleva el *software* más allá de los requisitos funcionales originales.
- *Prevención.* También llamado reingeniería del *software*, consiste en hacer cambios en los programas de computadora a fin de que se puedan corregir, adaptar y mejorar de manera más fácil.[3]

[3] En Apuntes de Informática II. Plan 2012. Facultad de Contaduría y Administración, UNAM.

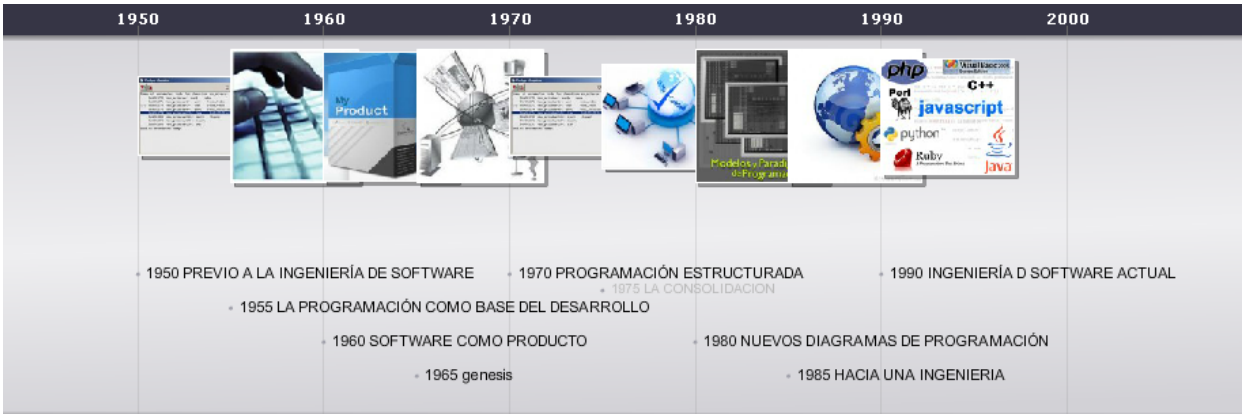


Fig. 1.1. Historia de la ingeniería del *software*

Fuente: http://goo.gl/GkcWRX

1.3. El *software* empresarial, su naturaleza y complejidad[4]

Según Peter G. W. Keende, en su obra *Shaping the Future*, una empresa capaz de cumplir sus objetivos en la sociedad actual debe ser "abierta" en el más amplio sentido de la palabra, para lo que debe tener una serie de características generales:

- Flexibilidad organizativa
- Adaptación al cambio
- Cobertura
- Extensiones interempresariales
- Cooperaciones y alianzas
- Procesos integrados
- Una gestión integrada y consistente

[4] Bravo, J. y Ortega, M. *Líneas de investigación en informática.* Disponible en http://goo.gl/RF8sqK. Recuperado: 31/05/2013.

Hoy día, las empresas compiten por satisfacer las necesidades de los clientes con la mayor calidad y agilidad, ofreciendo servicios y productos diferenciados y competitivos en calidad y precio. Para todo ello, resulta fundamental el alineamiento de las arquitecturas de los sistemas de información con las estrategias corporativas y los factores críticos de éxito, por lo que los sistemas de información deben constituir una herramienta eficaz para tener organizaciones flexibles, y contribuyan al rediseño de los procesos de negocio. Este diseño lleva a la agilización de las tomas de decisiones, lo que facilita la delegación de funciones y da lugar a organizaciones más planas, con procesos más productivos y motivadores.

Otros aspectos como la globalización y competencia a nivel mundial, entornos rápidamente cambiantes, movimientos hacia la integración o fusión (en el entorno bancario o seguros) y la descentralización (grandes empresas que se transforman en unidades independientes) también afectan sobremanera el ambiente del sistema informático.

1.4. Factores humanos que intervienen en la ingeniería del *software*

El ser humano está vinculado al proceso de ingeniería de *software*, desde su creación hasta su empleo. Así, al momento de realizar un proceso que involucre la generación del *software*, debemos acompañar a detalle la forma como se realizará la interacción.

El *software* forma parte de la "gestión del conocimiento", por eso también es llamado “procesos de negocio”: integra procesos tecnológicos y éstos a su vez se montan por el trabajo, la aplicación de conocimientos y la cooperación de

individuos organizados en maneras diversas. Fernando Sáez Vacas[5] define estos últimos elementos como “procesos psico-socio-económicos”. El autor establece una pirámide constructiva de "procesos psico-socio-económicos-procesos tecnológicos-procesos productivos", que emplea recursos cada vez menos relacionados con el capital, la tierra y las materias, y más vinculados al conocimiento y al capital humano e intelectual (tecnología, patentes, marcas, métodos, etcétera).

Al poner nombres a los distintos tipos de recursos que constituyen el capital intelectual explícito de una empresa que usa y gestiona esta pirámide de procesos, resulta bastante nemotécnico llamar *orgware* al conjunto de recursos que forman los procesos psicosocioeconómicos; *hardware* y *software* a los recursos para los procesos tecnológicos (o técnicos); y *produware* a los recursos aplicados en la creación y actualización de procesos productivos. Unos minutos de reflexión nos convencerán de que todos los recursos citados debería constituir el objeto de una compleja tarea de gestión del conocimiento, que va mucho más allá de una mera base de datos sobre cualificaciones del personal.

[5] Sáez Vaca, F. *Ingeniería de* software*. Factores económicos y humanos.* Disponible en http://goo.gl/tUiSfZ. Recuperado: 08/10/2013.

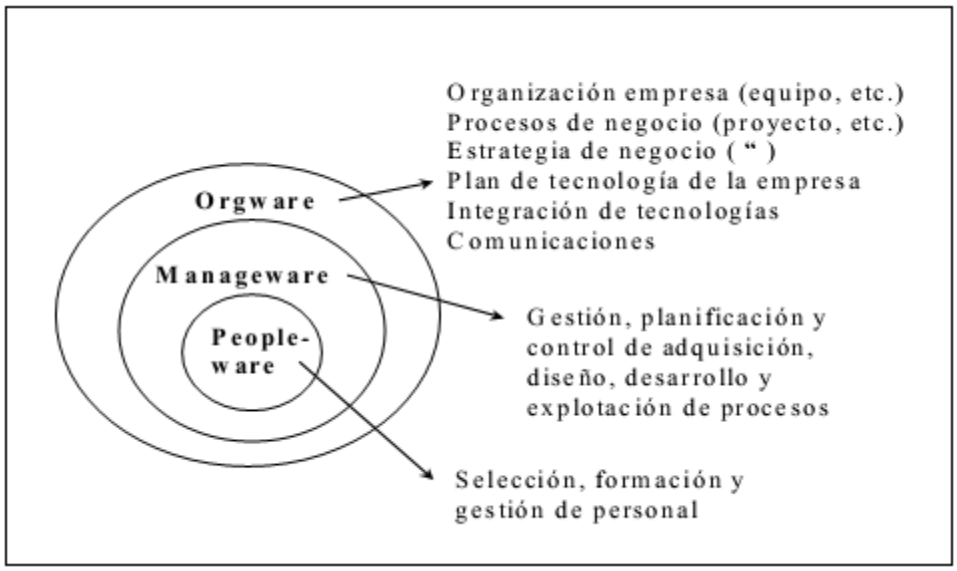


Figura 1.2. Diagrama *peopleware*.

Según como se le conciba, en el *orgware* pueden distinguirse convencionalmente otros dos aspectos, como el *manageware*, que se ocupa en concreto de tareas de gestión y planificación, compuestas por un variado abanico de técnicas y métodos; y el *peopleware*, cuya atención está más polarizada al factor humano propiamente dicho, tanto en su versión de grupo como de individuo.

Al *peopleware*, reducido a su faceta intelectual, se le podía llamar *brainware*, aunque ésta sería una visión reduccionista ignorante de sus facetas de ser social. El *peopleware*, por tanto, resumiría los conocimientos y técnicas enfocados a lo que, desde el punto de vista empresarial, es el capital humano (en términos prácticos, la selección, formación y gestión de personal).

1.5. Modelos para el desarrollo de sistemas

La ingeniería de *software* tiene varios modelos o paradigmas para el desarrollo de aplicaciones[6], que conllevan ventajas y desventajas.

Ejemplos[7]

- Modelo en cascada
- Modelo en espiral (modelo evolutivo)
- Modelo de prototipos
- Desarrollo iterativo e incremental

[6] Araya Fonseca, R. (2009). *Las mejores prácticas para el desarrollo de* software */ El proceso unificado de desarrollo.* En *Análisis y Diseño de Sistemas II* (2009). Fuente ULACIT.
[7] *Ingeniería de* software. Disponible en http://goo.gl/SU03nL. Recuperado: 08/10/2013.

Modelo en cascada

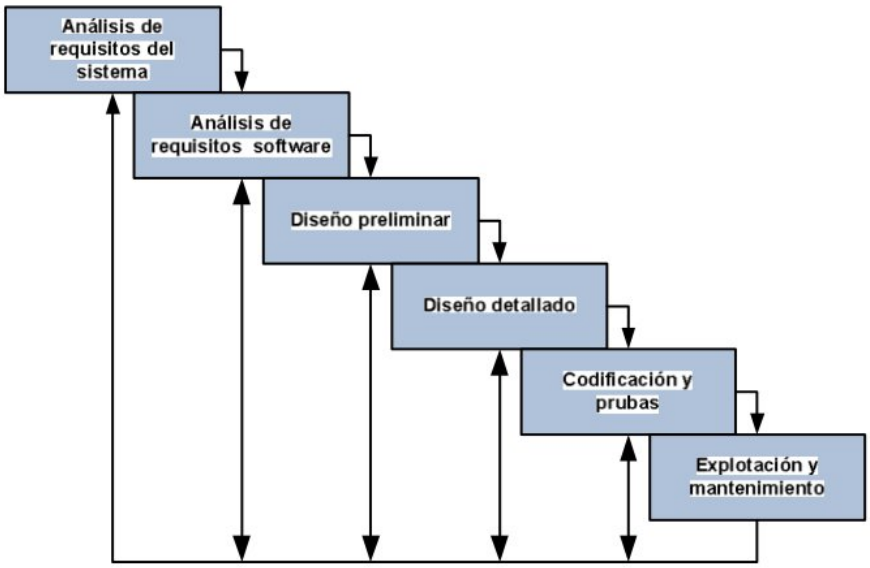


Figura 1.3. Modelo de cascada.

Como se puede observar en la figura anterior, el modelo en cascada es una metodología de desarrollo de *software* que parte de una fase de análisis de requisitos del sistema, donde se recopila toda la información referente a las características funcionales que tendrá el sistema, su ambiente de trabajo, los perfiles de usuarios y otros aspectos técnicos, para luego enfocarse a los requisitos propios del *software* derivados de los mismos requisitos del sistema en general. Después, se realizará un diseño a detalle del sistema, donde se especifiquen los módulos que lo integrarán, su funcionamiento individual y, en conjunto, las entradas y salidas de datos, entre otros. Una vez detallado y aprobado el diseño, se procede a la codificación o programación del sistema en un lenguaje de programación de alto nivel y a la construcción y captura de datos en un sistema administrador de base de datos.

Tanto el lenguaje de programación como el sistema manejador de base de datos y algunas otras herramientas de *software* necesarias para la construcción del sistema, quedan definidas a partir de la fase de análisis y diseño. Una vez codificado el sistema, se procede a realizar pruebas operativas, con el objetivo de verificar que los datos sean procesados y almacenados de forma correcta en el sistema y que su desempeño y funcionalidad estén acordes a las características establecidas por la empresa.

Realizadas las pruebas, se procede a la fase de explotación: poner a disposición de los usuarios el sistema para que lo empleen y desempeñen sus labores de manera más eficiente. Hecho esto, se debe efectuar un mantenimiento periódico con el objetivo de conservar el funcionamiento del sistema de forma adecuada y, llegado el momento, proceder a su sustitución o realizar mejoras al mismo. Para alcanzar el funcionamiento deseado del sistema de información, el modelo en cascada permite regresar a cualquiera de sus fases.

Modelo incremental

Ocupa elementos del modelo en cascada aplicados en forma iterativa; es decir, cada parte del sistema es construida de forma individual, entregando en cada pasada un producto operacional que puede evaluarse y, posteriormente, integrarse a los módulos siguientes desarrollados en cada iteración. Es útil cuando no se cuenta con todo el personal necesario para realizar el proyecto o habilitar líneas paralelas de desarrollo.

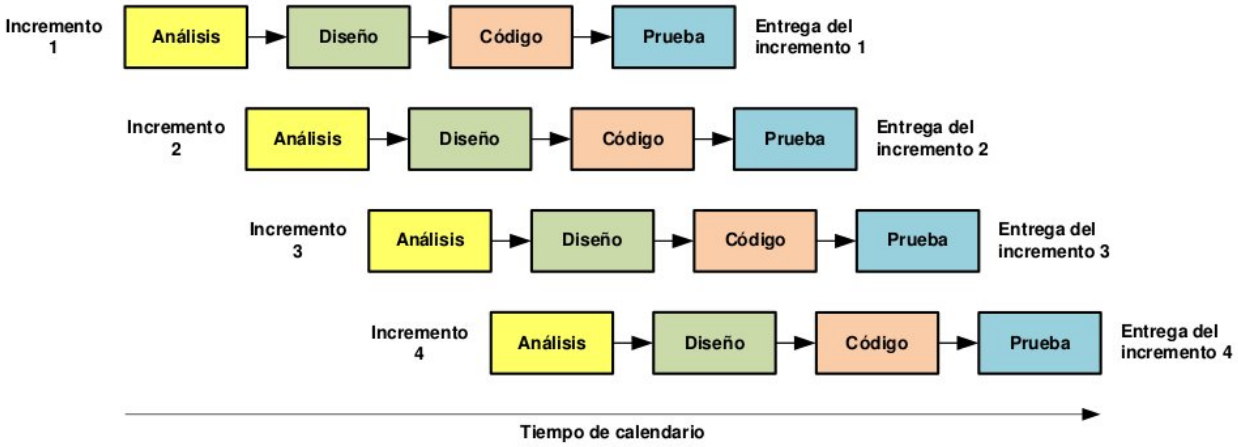


Figura 1.4. Modelo incremental[8].

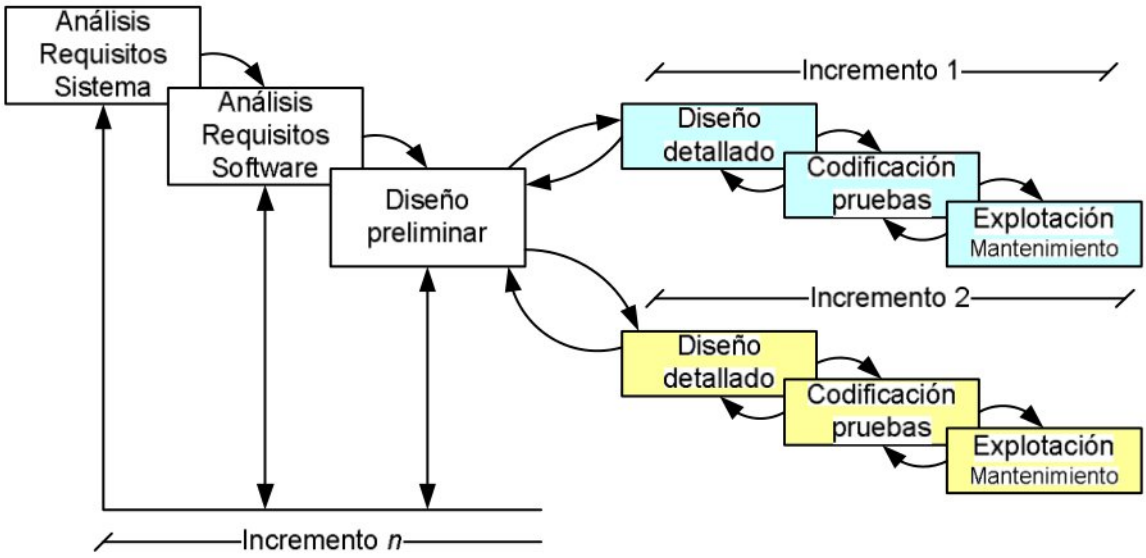


Figura 1.5. Procedimiento del modelo incremental.[9]

[8] En *Análisis y diseño de sistemas-Modelos para el desarrollo de* software. Disponible en **http://goo.gl/BbUkZ1**. Recuperado: 08/10/2013.

[9] En *Análisis y diseño de sistemas-Modelos para el desarrollo de* software. Disponible en http://goo.gl/iyAiHq. Recuperado: 08/10/2013.

Como se observa en la figura anterior, las fases de cada incremento son coincidentes con las del modelo en cascada, pero aplicadas a un proceso de desarrollo tipo modular.

Modelo en espiral

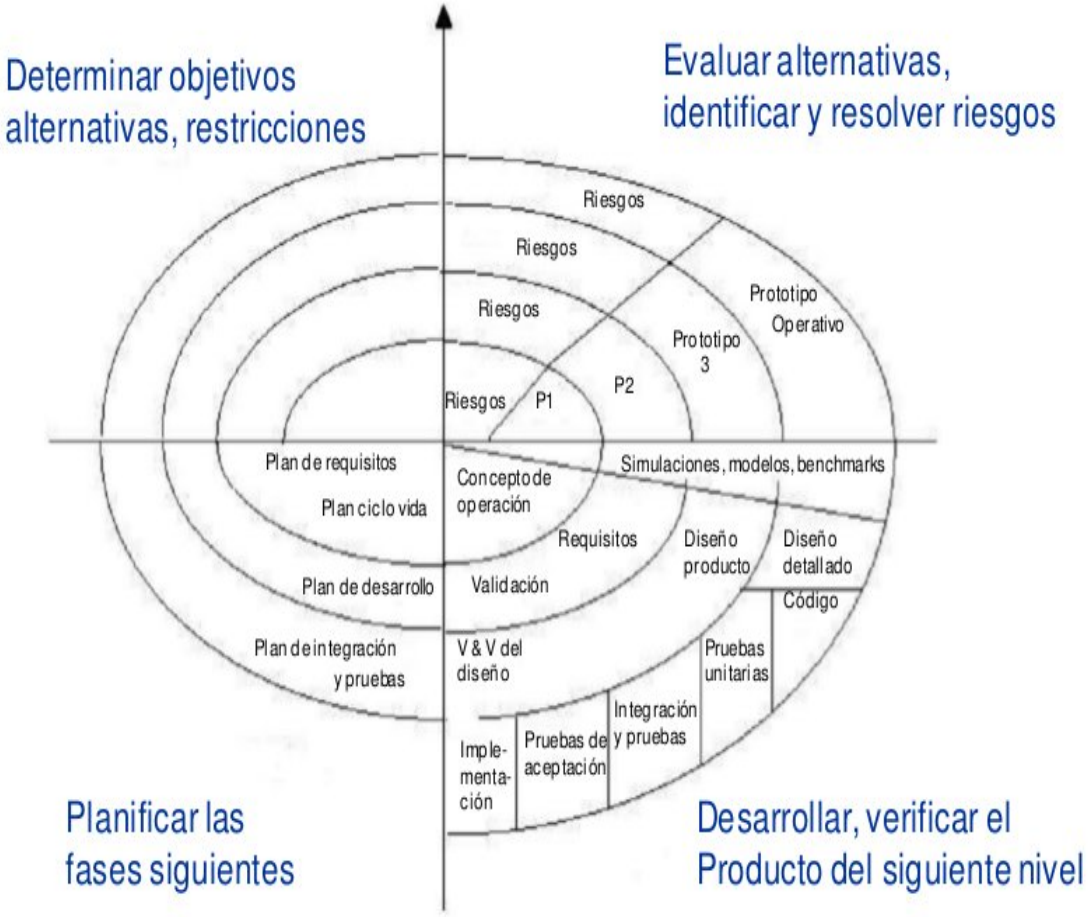


Figura 1.6. Modelo en espiral.[10]

[10] En *Análisis y diseño de sistemas-Modelos para el desarrollo de* software. Disponible en http://goo.gl/dXMzuw. Recuperado: 08/10/2013.

El modelo en espiral se basa en el desarrollo de prototipos funcionales del sistema, donde para la obtención de cada prototipo se parte de una planificación que integra tanto el modelo de desarrollo a seguir como la forma de integración de los prototipos, requisitos del siguiente prototipo, etcétera. Luego, se pasa a una fase donde se establecen los objetivos, alternativas de desarrollo y restricciones para la construcción del prototipo. Después, se procede a evaluar los riesgos en el desarrollo del prototipo y las alternativas de solución. Establecido lo anterior, se procede a la construcción del prototipo, su validación y pruebas, para su evaluación posterior por parte del dueño de la empresa y los usuarios. Realizadas estas fases, el modelo se reinicia hasta obtener el producto final.

Modelo de prototipos

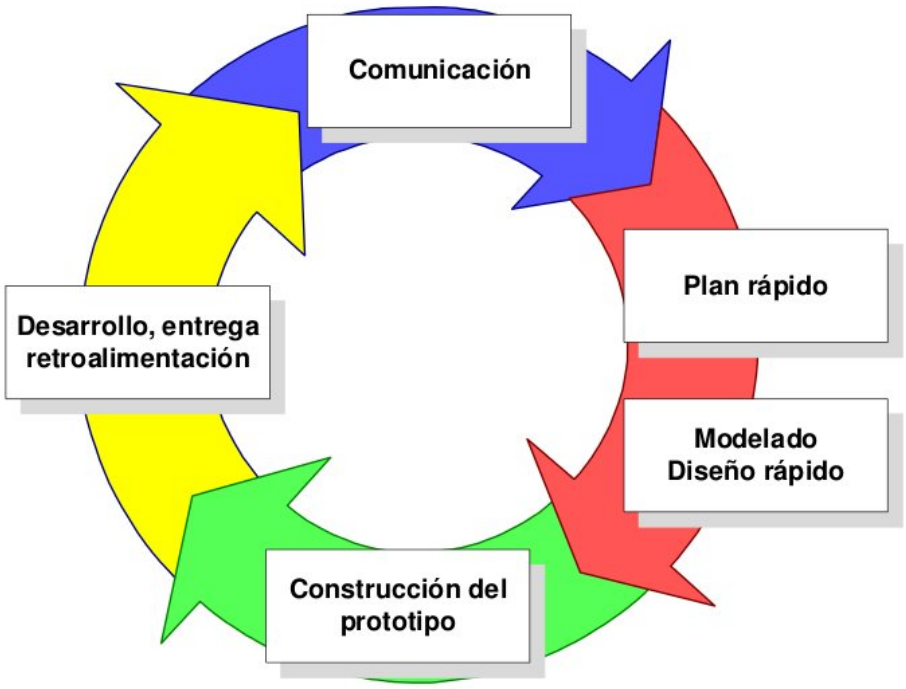


Figura 1.7. Modelo de prototipos.[11]

[11] En *Análisis y diseño de sistemas-Modelos para el desarrollo de* software. Disponible en http://goo.gl/6dlRxl. Recuperado: 08/10/2013.

El modelo de prototipos es muy similar al de desarrollo en espiral, ya que su principal objetivo es ir desarrollando un prototipo funcional del sistema de forma incremental siguiendo los pasos del modelo incremental, evaluarlo y hacerlo crecer en cada pasada. El modelo es un ciclo finito de iteraciones, que culmina hasta que el sistema ha sido desarrollado en su totalidad.

1.6. Los sistemas de información como una ventaja competitiva en las organizaciones

El papel estratégico de los sistemas de información incluye el uso de la tecnología de información para desarrollar productos, servicios y capacidades que dan a una empresa ventajas estratégicas sobre las fuerzas competitivas que ésta enfrenta en el mercado global.

El éxito de la supervivencia a largo plazo radica en la aplicación de estrategias para confrontar cinco fuerzas competitivas que dan forma a la estructura de la competencia en su industria:

1. Competencia directa
2. Nuevos participantes
3. Productos sustitutos
4. Habilidades de negociación de los clientes
5. Habilidades de negociación de los proveedores

Otros ejemplos de tipos de negociación pueden ser consultados en la bibliografía.[12]

[12] Merino, S. *Apuntes de Ingeniería de Sistemas: III Parte. Sistemas de información para obtener ventaja competitiva.* Disponible en http://goo.gl/h8U8FL. Recuperado: 08/10/2013.

Papeles estratégicos de los sistemas de información

¿De qué manera se aplican los conceptos de estrategia competitiva anteriores al papel estratégico de los sistemas de información en una organización? ¿Cómo usan los gerentes las inversiones en tecnología de información para respaldar directamente las estrategias competitivas de una empresa? Estas preguntas pueden responderse en términos de los papeles estratégicos clave que los sistemas de información desempeñan en una empresa.

A continuación, se resume cómo puede utilizarse la tecnología de información para implementar una variedad de estrategias competitivas, que incluyen no sólo las cinco estrategias competitivas básicas, sino también otras maneras como las empresas usan los sistemas de información estratégicamente para lograr un margen competitivo.

Reducción de costos

- Utilizar TI para reducir de manera sustancial el costo de los procesos empresariales y también los costos de clientes y proveedores.

Diferenciación

- Desarrollar nuevas características de TI para diferenciar productos y servicios.
- Utilizar características de TI para reducir las ventajas de diferenciación de los competidores.
- Utilizar características de TI para centrar los productos y servicios en nichos de mercado seleccionados.

Innovación

- Crear nuevos productos y servicios que incluyan componentes de TI.
- Realizar cambios radicales en los procesos empresariales con TI.
- Desarrollar nuevos mercados o nichos de mercado únicos con TI.

Promoción

- Utilizar TI para manejar la expansión empresarial regional y global.

Crecimiento

- Utilizar TI para diversificarse e integrarse en otros productos y servicios.

Desarrollo de alianzas

- Utilizar TI para crear organizaciones virtuales de socios comerciales.

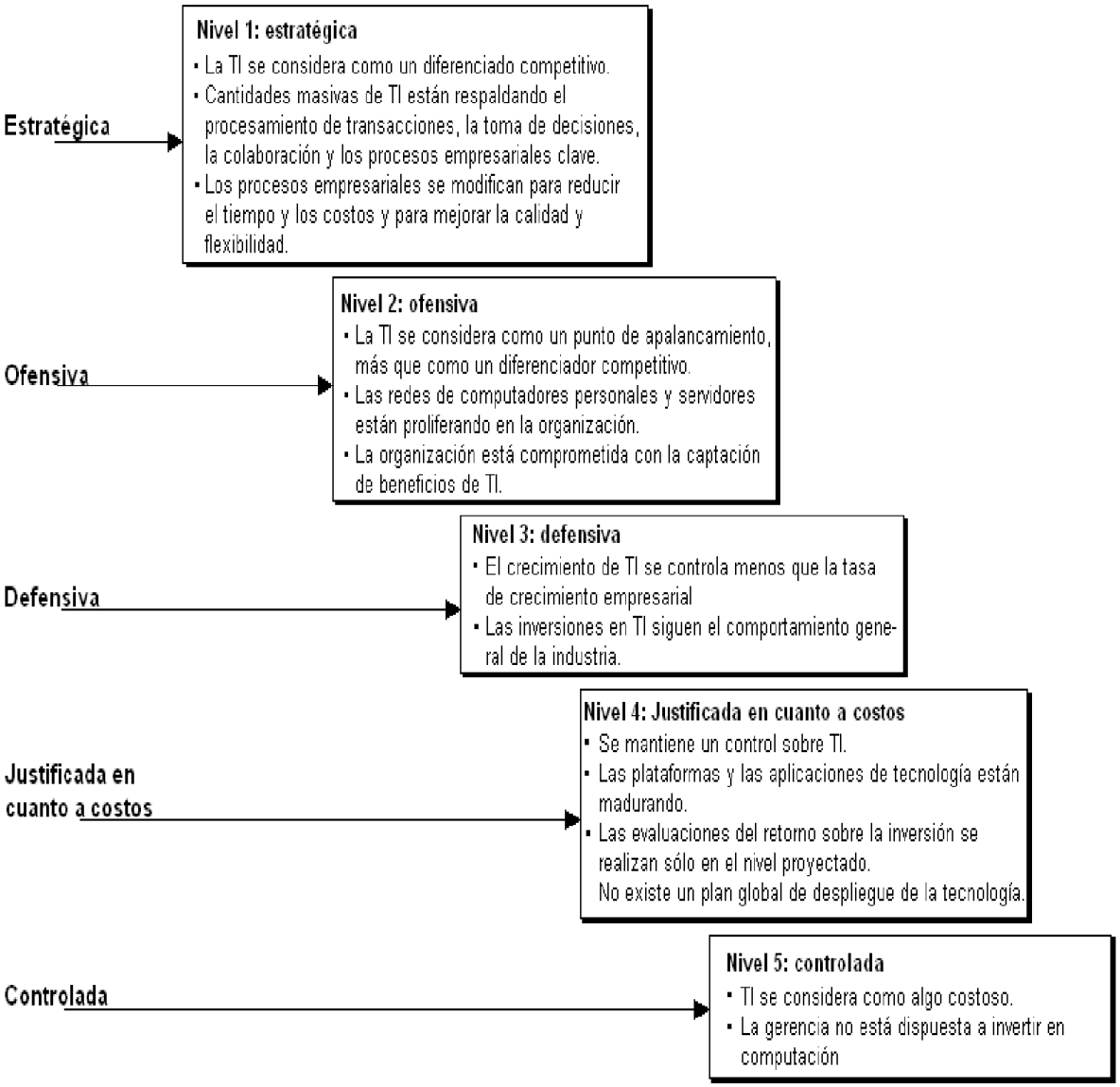


Figura 1.8. Papeles estratégicos de los sistemas de información.[13]

[13] Merino, S. *Apuntes de Ingeniería de Sistemas: III Parte. Sistemas de información para obtener ventaja competitiva.* Disponible en http://goo.gl/8lXnsT. Recuperado: 08/10/2013.

RESUMEN DE LA UNIDAD

Los sistemas de información empresarial permiten a las empresas desarrollar sus procesos de manera más eficiente al facilitar el intercambio de información continuo entre las áreas que la integran, lo que ayuda en la toma de decisiones y el procesamiento de la información generada el interior y exterior de la empresa.

Los sistemas empresariales no son otra cosa que *software* a la medida de las necesidades de una organización. Por ello, para su construcción, es necesario partir desde los objetivos mismos de la empresa, para que el *software* resuelva las necesidades para el cual fue diseñado. Como en la construcción de todo *software* de calidad, es necesario seguir una cierta metodología de desarrollo, que parte desde el análisis de los requisitos del sistema en general, hasta su construcción y puesta en marcha. En este orden, es posible ajustarse a diversas metodologías establecidas dentro de la ingeniería de *software* que permiten desarrollar de forma efectiva el sistema de información deseado.

El factor humano es un aspecto fundamental para el desarrollo de los sistemas empresariales, ya que el capital humano de una empresa se ve reflejado en los sistemas de la misma. Los seres humanos plasman su conocimiento en el desarrollo de los sistemas de información, que se concreta en mejoras de los procesos productivos, la toma de decisiones y, al final, la generación de nuevo conocimiento para la empresa.

Por todo esto, los sistemas empresariales representan una gran ventaja competitiva para las empresas, a través de ellos, pueden obtener información valiosa para analizar su entorno y establecer estrategias adecuadas para mantener una posición de liderazgo en el mercado. Así, los sistemas de información son creados con fines estratégicos, con el objetivo de explotar las fortalezas de la empresa y mejorar su competitividad.

GLOSARIO DE LA UNIDAD

Hardware

Componentes físicos de una computadora. Todo el soporte físico de una computadora como disco duro, gabinetes, monitores, unidades de disco, etcétera.

Ingeniería

Disciplina que integra conocimientos de diferentes áreas como la matemática, física, electrónica, entre otras, para la solución de problemas complejos a partir de metodologías de análisis, diseño e implementación.

Ingeniería del *software*

Aplicación y estudio de los procesos sistemáticos y disciplinados para el desarrollo, operación y mantenimiento de *software*.

Programación

Proceso de diseñar, codificar, depurar y mantener el código fuente de programas computacionales. El propósito de la programación es crear programas que exhiban un comportamiento deseado en un ordenador.

Prototipo

Versión preliminar de un sistema que sirve de modelo para fases posteriores.

Requisito

Condición o facultad que necesita un usuario para resolver un problema. O condición o facultad que debe poseer un sistema o un componente de un sistema para satisfacer una especificación, estándar, condición de contrato u otra formalidad impuesta documentalmente. También es el documento donde se recogen los puntos anteriores.

Sistema

Conjunto de procesos o elementos interrelacionados con un medio para formar una totalidad encauzada hacia un objetivo común.

Sistema de información (informático)

Conjunto organizado de elementos que pueden ser personas, datos, actividades o recursos materiales en general. Estos elementos interactúan entre sí para procesar información y distribuirla de manera adecuada en función de los objetivos de una organización.

Software

Todo el conjunto intangible de datos y programas de la computadora.

Validación

Confirmación, mediante examen y aportación de pruebas objetivas, de que se cumplen los requisitos concretos para un uso determinado. Responde a la pregunta *¿estamos construyendo el producto correcto?*

Verificación

Confirmación, mediante examen y aportación de pruebas objetivas, de que se cumplen los requisitos específicos. Responde a la pregunta *¿estamos construyendo correctamente el producto?*

ACTIVIDADES DE APRENDIZAJE

ACTIVIDAD 1

Realiza una investigación sobre tres modelos de desarrollo de *software* diferentes a los mencionados en este material didáctico. Después, elabora un cuadro comparativo de los modelos investigados y los modelos presentados en el material, con las fases que los integran, características particulares y recomendaciones para su empleo.

ACTIVIDAD 2

Discute con tus compañeros el tema "Modelos de desarrollo de *software* y sus características", a partir de las preguntas siguientes:

1. ¿Por qué es necesario seguir un modelo o metodología de desarrollo de *software*?
2. ¿Cuáles son sus características generales?
3. ¿Se obtiene *software* de calidad al seguir una metodología?

ACTIVIDAD 3

Elabora un ensayo de dos páginas, donde reflexiones sobre el impacto del factor humano en el desarrollo de los sistemas empresariales. Enfoca tu reflexión a la manera como las empresas transforman el conocimiento obtenido de su capital humano en sistemas empresariales y su empleo dentro de las empresas.

ACTIVIDAD 4

En el foro de discusión, reflexiona con tus compañeros sobre los sistemas empresariales como ventajas competitivas en las empresas, a partir de estas preguntas:

1. ¿Qué es una ventaja competitiva?
2. ¿Cómo ayudan los sistemas de información a las empresas?
3. ¿Cómo se convierten en ventajas competitivas los sistemas empresariales?

Coloca tu respuesta en el foro y realiza un comentario sobre los aportes de tus compañeros.

ACTIVIDAD 5

Escribe tres ejemplos del empleo de sistemas empresariales como parte de una ventaja estratégica en una empresa.

CUESTIONARIO DE REFORZAMIENTO

Responde las siguientes preguntas.

1. ¿Qué es un sistema de información empresarial?
2. ¿Cuáles son los objetivos del *software* empresarial?
3. ¿Cuáles son las fases o pasos principales de la ingeniería de *software*?
4. Menciona cuatro modelos de desarrollo de *software* empleados en la ingeniería de *software*.
5. Explica brevemente cuáles son las similitudes en los modelos de desarrollo de *software*.
6. ¿Qué es la pirámide procesos psico-socio-económicos-procesos tecnológicos-procesos productivos?
7. ¿Cómo interviene el factor humano en la pirámide de procesos?
8. ¿Qué es un modelo de desarrollo de *software*?
9. ¿Qué es una ventaja competitiva?
10. ¿Cómo ayuda el *software* empresarial a consolidar las ventajas competitivas y estratégicas de una empresa?

LO QUE APRENDÍ

Retoma la actividad “Lo que sé”, y complementa tu mapa relacionándolo adicionalmente con las ventajas competitivas y estratégicas de una empresa.

EXAMEN DE AUTOEVALUACIÓN

Relaciona ambas columnas.

1	Modelo de desarrollo de *software* que parte de la fase de análisis de requisitos del sistema y continúa de forma secuencial hasta la fase de explotación y mantenimiento; permite regresar a cualquiera de sus fases si es necesario para alcanzar el funcionamiento deseado del sistema de información.	A. Definición
2	Sistemas enfocados a la administración del flujo de datos dentro de una organización; su principal objetivo es ofrecer información con prontitud, confianza y asertividad a todas sus dependencias.	B. Estrategia de innovación
3	Fase de la ingeniería de *software* que se centra en el *cómo* Es decir, se intenta definir cómo han de diseñarse las estructuras de datos, construirse la función como una arquitectura del *software*, implantarse detalles sobre los procedimientos, caracterizarse las interfaces, traducirse el diseño en el lenguaje de programación y realizarse las pruebas.	C. Mantenimiento
4	Modelo de desarrollo basado principalmente en el desarrollo de prototipos funcionales del sistema, donde para la obtención de cada prototipo se parte de una planificación que integra tanto el modelo de desarrollo a seguir, como la forma de integración de los prototipos, los requisitos del siguiente prototipo, etcétera.	D. Ingeniería de *software*

5	Estrategia para encontrar nuevas maneras de hacer negocios. Esto puede comprender el desarrollo de productos y servicios únicos, o ingresar a mercados o nichos de mercado exclusivos.	E. Modelo en cascada
6	De acuerdo con la pirámide propuesta por Fernando Sáez, Capa, cuya atención está más polarizada al factor humano propiamente dicho, tanto en su versión de grupo como de individuo.	F. *Orgware*
7	Según Fernando Sáez, capa donde pueden distinguirse convencionalmente aspectos como el *manageware*, que se ocupa específicamente de tareas de gestión, planificación y gestión.	G. Modelo en espiral
8	Fase de la ingeniería de *software* centrada en el *qué.* Es decir, intenta identificar qué información ha de ser procesada, qué función y rendimiento se desea, qué comportamiento del sistema, qué interfaces van a ser establecidas, qué restricciones de diseño existen y qué criterios de validación se necesitan para definir un sistema correcto.	H. Desarrollo
9	Conjunto de métodos, herramientas, técnicas y procesos necesarios para la generación de programas empleados en entornos informatizados.	I. Sistema de información empresarial
10	Fase de la ingeniería de *software* centrada en el cambio que va asociado a la corrección de errores, a las adaptaciones requeridas a medida que evoluciona el entorno del *software* y a modificaciones debidas a las mejoras producidas por los requisitos cambiantes de las necesidades del cliente.	J. *Peopleware*

MESOGRAFÍA

BIBLIOGRAFÍA RECOMENDADA

Autor	Capítulo	Páginas
Alonso	10	110-116
Pressman	2	23-30

BIBLIOGRAFÍA BÁSICA

Alonso Martínez, Segovia. *Introducción a la ingeniería del software.* España: Delta Publicaciones Universitarias, 2005, 537 pp.

Pressman, Roger S. *Ingeniería del* software. *Un enfoque práctico*, 5.ª ed. España: McGraw-Hill, 2002, 601 pp.

BIBLIOGRAFÍA COMPLEMENTARIA

1. Bardou, Louis. *Mantenimiento y soporte logístico de los sistemas informáticos.* México: Alfa Omega-Marcombo, 2004, 292 pp.
2. Bochinno, William A. *Sistemas de Información para la administración, técnicas e instrumentos.* México: Trillas, 2002, 403 pp.
3. Bonsón, Enrique. *Tecnologías inteligentes para la gestión empresarial*. México: Alfa Omega-Rama, 2002, 258 pp.
4. Cornella, Alfons. *Información digital para la empresa, una introducción a los servicios de información electrónica.* México: Alfa Omega-Marcombo, 2004, 196 pp.

5. Lardent, Alberto R. *Sistemas de información para la gestión empresarial, procedimientos, seguridad y auditoría*. Buenos Aires: Pearson Education-Prentice Hall, 2001.443 pp.
6. Levine, Guillermo. *Computación y programación moderna*. México: Addison Wesley, 2000, 609 pp.
7. Long, Nancy y Larry Long. *Introducción a las computadoras y a los sistemas de información, edición Internet.* México: Prentice Hall, 1999, 416 pp.
8. McLeod, Raymond Jr. *Sistemas de información gerencial,* 7.ª ed. México: Prentice Hall, 1999, 688 pp.
9. Oz, Effy. *Administración de sistemas de información,* 2.ª ed. México: Thomson-Learning, 2001, 578 pp.
10. Peña R., Baeza-Yates, R. y Rodríguez, J. *Gestión digital de la información. de bits a bibliotecas digitales y la web.* México: Alfa Omega-Rama, 2004, 464 pp.
11. Piattini, Mario; J. Antonio Calvo-Manzano, Joaquín Cervera y Luis Fernández. *Análisis y diseño detallado de aplicaciones informáticas de gestión.* México: Alfa Omega-Rama, 2004, 728 pp.
12. Stair, Ralph M. *Principios de sistemas de información*, 4.ª ed. México: Thomson Learning, 2002, 692 pp.
13. Walker, D. W. *Sistemas e información para la administración.* México: Alfa Omega-Marcombo, 2001, 360 pp.

SITIOS ELECTRÓNICOS

Dirección electrónica	Descripción
http://goo.gl/QuiIN4	Ingeniería de *software.*
http://goo.gl/0CmCDK	Introducción a la ingeniería del *software.*

UNIDAD 2. ANÁLISIS DE SISTEMAS

OBJETIVO ESPECÍFICO

Al finalizar la unidad, el alumno identificará los procesos necesarios para realizar el análisis a detalle de un sistema de información empresarial.

INTRODUCCIÓN

Cualquier modelo de desarrollo de sistemas debe incluir una etapa de análisis, en la cual los desarrolladores o ingenieros de *software* recopilarán la información necesaria para poder comenzar con su construcción, y realizarán un bosquejo inicial de la forma como estará estructurado. En esta línea, la unidad se concentra en los conceptos principales que integran la fase de análisis del sistema de acuerdo con los siguientes puntos:

- *Objetivos del análisis.* Hacen referencia a lo que se desea realizar con el sistema; deben estar acordes con los objetivos organizacionales.

- *Estudio de viabilidad.* Determina si el sistema en cuestión es posible de realizar.

- *Estudio económico técnico.* Determina si se cuenta con los recursos financieros y tecnológicos adecuados para desarrollar el sistema.

- *Conceptos de calidad.* Hacen referencia a aquellos criterios que deben ser considerados para obtener un sistema de información adecuado que cumpla con todas sus especificaciones.

- *Análisis de requisitos del sistema.* En este proceso se establecen las características funcionales y no funcionales que debe tener el sistema.

- *Especificaciones funcionales del sistema.* Proceso que parte del análisis de los requisitos del sistema hasta el modelado del mismo, dando como resultado una visión general de su estructura interna.

- *Interfaces del sistema.* Proceso donde se representan los módulos a través de los cuales los usuarios podrán interactuar con el sistema.

LO QUE SÉ

Elabora un cuadro sinóptico de los procesos que, de acuerdo con tu conocimiento, integran la fase de análisis de un sistema de información.

TEMARIO DETALLADO (12 horas)

2.1. Objetivos del análisis
2.2. Estudio de viabilidad
2.3. Estudio económico y técnico
2.4. Conceptos de calidad (CMM) en el análisis de sistemas
2.5. Análisis de requisitos del sistema
2.6. Especificación funcional del sistema
    2.6.1. Identificación y definición de requisitos
    2.6.2. Diseño del modelo lógico actual
    2.6.3. Estudio de alternativas de construcción
2.7. Especificación funcional del sistema
    2.7.1. Construcción del modelo de procesos del nuevo sistema
    2.7.2. Construcción del modelo de datos del nuevo sistema
    2.7.3. Elaboración de análisis detallado del nuevo sistema
2.8. Interfaz del sistema
    2.8.1. Definición de interfaces con el usuario
    2.8.2. Complementar las especificaciones del sistema
    2.8.3. Complementar las especificaciones de entrega

2.1. Objetivos del análisis

El objetivo del análisis de sistemas es identificar con precisión las necesidades de información de una organización y establecer la alternativa de solución más conveniente para satisfacerla. Para ello se recomiendan los siguientes pasos:

1. Obtener los requisitos del cliente para el sistema.
2. Identificar escenarios o casos de uso.
3. Seleccionar clases y objetos usando los requisitos básicos de programación.
4. Identificar los atributos y operaciones para cada objeto del sistema.
5. Definir estructuras y jerarquías

2.2. Estudio de viabilidad

Todo proyecto, sea de *software* o no, debe realizar un estudio de viabilidad (también conocido como factibilidad) a fin de seleccionar la metodología más adecuada que satisfaga los requisitos del sistema, tanto en su parte técnica, como en su rendimiento. El objetivo principal es determinar si será económica y técnicamente factible, para ello se debe analizar el problema y recopilar toda la información pertinente relacionada con el producto, como los diferentes elementos de datos que se pueden introducir en el sistema, el procesamiento necesario para llevar a cabo estos datos, los datos de salida necesarios a ser producidos por el sistema, así como diversas restricciones sobre su comportamiento.

Factibilidad técnica

Se refiere a la especificación de equipos y *software* que van a satisfacer de forma exitosa los requerimientos del usuario. Aunque las necesidades técnicas del sistema dependerán de cada proyecto, en términos generales incluyen lo siguiente[14]:

- Facilidad para producir resultados en un tiempo determinado.
- Tiempo de respuesta en condiciones específicas.
- Capacidad para procesar un volumen de transacciones a una velocidad determinada.
- Facilidad para comunicar datos a lugares distantes.

Viabilidad económica

El análisis económico es la técnica más utilizada para evaluar la eficacia de sistema a ser desarrollado. Conocido también como análisis de costo/beneficio, el procedimiento consiste en determinar los beneficios y ahorros que se esperan de un sistema a desplegar y compararlos con sus costos. Si los beneficios superan a los costos, se toma una decisión de diseñar e implementar el sistema; de lo contrario, tendrá que ser realizada una justificación adicional o alternativa en el sistema a generar para que pueda ser aprobada.

[14] Basado en el texto *Feasibility study-software engineering*. Disponible en http://goo.gl/JNFdj7. Recuperado: 08/10/2013.

Factibilidad operacional

Esto se relaciona principalmente con los aspectos de la organización donde será implantado el sistema y su interacción con los usuarios. En este caso, los puntos a considerar son los siguientes:

- ¿Qué cambios serán traídos con el sistema?
- ¿Qué partes de la organización serán afectadas?
- ¿Qué nuevas habilidades serán requeridas?
- ¿Los miembros del personal cuentan con estas habilidades?

Este estudio de viabilidad se lleva a cabo por un pequeño grupo de personas familiarizadas con la parte técnica del sistema.

Etapas

- *Establecimiento del alcance del sistema*. Se establecen las características deseables del sistema a construir, así como las áreas involucradas en su desarrollo.

- *Estudio de la situación actual*. Ofrece una descripción de la forma como las áreas involucradas en el sistema se encuentran realizando sus actividades.

- *Definición de requisitos del sistema.* Se describe a detalle el funcionamiento que debe tener el sistema, así como la manera como interactuará con los diversos usuarios del mismo y su entorno.

- *Estudio de alternativas de solución.* Se plantean diversas propuestas que pueden responder a las necesidades a las cuales dará solución el sistema.

- *Valoración de las alternativas.* Establecidas las propuestas de solución, se debe evaluar a detalle cada una de ellas para determinar cuál será la que se adapte mejor a las necesidades planteadas.

- *Selección de la solución.* Evaluadas las alternativas, se elegirá una o varias que darán vida al sistema de información.

2.3. Estudio económico y técnico

Factibilidad técnica

El estudio de viabilidad técnica compara el nivel de la tecnología disponible en la empresa de desarrollo de *software* con el nivel de la tecnología necesaria para el desarrollo del producto (en este caso, el nivel de la tecnología consiste en el lenguaje de programación, recursos de *hardware*, *software*, otras herramientas, etcétera). Los resultados obtenidos son la base para determinar sobre si continuar o abandonar el proyecto, si hay riesgos de que no funcione, no tenga el rendimiento deseado, o si las piezas no encajan perfectamente unas con otras.[15]

Se recomiendan técnicas para facilitar las especificaciones de la aplicación (TFEA[16]), las cuales, con variaciones según cada proyecto, tienen las siguientes directrices:

- La reunión se celebra en un lugar neutral y acuden tanto clientes como desarrolladores.

[15] *Análisis de sistemas de computación*. Disponible en http://goo.gl/auhq2F. Recuperado: 08/07/2013.
[16] Dos de los enfoques más populares de TFEA son Desarrollo Conjunto de Aplicaciones (JAD), creado por IBM; y The METHOD, de Performance Resources, Inc., Falls Church, VA.
Fuente: Apuntes de Ingeniería de Software, Unidad 4. Desarrollados por alumnos del Instituto Tecnológico de Ciudad Juárez, Chihuahua. Disponible en http://goo.gl/1RfpEB. Recuperado: 08/07/2013.

- Se establecen normas de preparación y participación.
- Se sugiere una agenda lo suficientemente formal como para cubrir todos los puntos importantes; pero a la vez lo suficientemente informal como para animar el libre flujo de ideas.
- Un coordinador (que puede ser un cliente, un desarrollador o un tercero) que controle la reunión.
- Se usa un mecanismo de definición (hojas de trabajo, gráficos, carteles o pizarras).
- El objetivo es identificar el problema, proponer elementos de solución, negociar diferentes enfoques y especificar un conjunto preliminar de requisitos de la solución en una atmósfera que permita alcanzar el objetivo.

Estudio económico

El estudio de viabilidad económica evalúa el costo del desarrollo de *software* contra los ingresos o beneficios obtenidos una vez finalizado de forma exitosa el proyecto (véase el punto 2.2). Al evaluar la viabilidad económica de una alternativa de aplicación, la pregunta básica a responder es *¿el proyecto tiene viabilidad financiera?* Esto se hace mediante la realización de un análisis de costo/beneficio, donde se compara el costo total real/de la aplicación a sus beneficios financieros completos/real. Las alternativas deben ser evaluadas en función de su contribución al flujo neto de efectivo, la cantidad en que los beneficios superan a los costos, ya que el principal objetivo de las inversiones es mejorar el rendimiento general de la organización.

La siguiente tabla muestra algunos de los costos y beneficios potenciales que pueden ser adquiridos en un proyecto de *software*. Aunque la lista no es exhaustiva, proporciona una indicación de la variedad de factores que se deben tener en cuenta al evaluar la viabilidad económica de una aplicación. La tabla incluye tanto los factores

cualitativos, costos o beneficios que son de carácter subjetivo, como los factores cuantitativos, costos o los beneficios monetarios.

Costos y beneficios potenciales de un proyecto de *software*

Tipo	Costos potenciales	Beneficios potenciales
Cuantitativo	• Actualizaciones de *hardware/software.* • Costo laborales (salarios + prestaciones). • Costos de soporte de la aplicación. • Costos operativos esperados. • Gastos de formación para que los usuarios aprendan la aplicación. • Gastos de formación para capacitar a los desarrolladores en nuevas o actualizadas tecnologías.	• Reducción de costos de operación. • Reducción de costes de personal al reducir personal. • Aumento de los ingresos por las ventas adicionales de sus productos o servicios de las organizaciones.
Cualitativo	• Aumento de insatisfacción de los empleados por el miedo al cambio. • Percepción pública negativa de despidos como el resultado de la automatización.	• Decisiones mejoradas como resultado del acceso a información precisa y oportuna. • Aumento o generación de una nueva barrera dentro de la industria para mantener la competencia de su mercado. • Percepción pública positiva acerca de que la

		organización es una empresa innovadora.

Análisis cuantitativo de costo/beneficio

Para realizar un análisis de costo/beneficio cuantitativo es necesario identificar los costos iníciales del desarrollo, los costos esperados de la operación y el soporte a la aplicación, así como los beneficios monetarios futuros esperados por la utilización de la aplicación. Debido a que los costos y beneficios se acumulan en diferentes momentos, algunos de inmediato y otros en el futuro, es necesario convertir los costos a los valores actuales para que se puedan comparar de forma adecuada.

Ejemplo

Concepto	*Año 1*	*Año 2*	*Año 3*	*Año 4*	*Año 5*
Costos	$ 18,055,858.00	$ 21,947,589.00	$ 26,435,582.00	$ 31,987,055.00	$ 38,704,338.00
Factor	0.83	0.68	0.56	0.47	0.39
Valor presente	$ 14,986,362.14	$ 14,924,360.52	$ 14,803,925.92	$ 15,033,915.85	$ 15,094,691.82
Beneficios	$ 5,250,899.00	$ 6,353,589.00	$ 7,687,842.00	$ 9,302,289.00	$ 11,255,770.00
Factor	0.83	0.68	0.56	0.47	0.39
Valor presente	$ 4,358,246.17	$ 6,391,394.65	$ 9,774,045.13	$ 14,704,258.30	$ 21,570,278.00

Análisis de cualitativo de costo/beneficio

Hay dos líneas de pensamiento sobre el análisis costo/beneficio cualitativo. La primera estrategia es mantener las cosas simples y aplicar el sentido común: si el proyecto parece una buena idea, lo más probable es que lo sea. La segunda línea de pensamiento es que se pueden cuantificar los aspectos cualitativos de un proyecto y, por tanto, realizar una comparación justa.

Pasos para cuantificar los factores cualitativos.

1. *Identificar los factores cualitativos.* Lluvia de ideas con varias personas.
2. *Cuantificar la importancia de cada factor a su organización*. Por ejemplo, dar a cada factor una calificación de uno a cinco, donde cinco es la más importante.
3. *Tasar numéricamente cada alternativa respecto a cada factor cualitativo.* Por ejemplo, la tasa de cada alternativa en una escala de cero a diez, donde diez es la calificación más alta posible.
4. *Multiplicar la ponderación importancia de la clasificación de cada alternativa.*
5. *Calcular la puntuación global para cada alternativa sumando las puntuaciones individuales.*

Mejores prácticas

Al determinar la viabilidad económica de una alternativa, hay varias acciones importantes a realizar:

1. *Asignar todos los costos.* Si su proyecto requiere actualizaciones de *hardware / software* que otras aplicaciones también demandan, entonces no se deberá asumir el costo total de la actualización. En este caso, es necesario negociar sólo parte de la actualización con la alta dirección.

2. *Asignar los beneficios de manera justa.* Muchos beneficios se pueden lograr a través de la mejora de procesos de negocio sin la necesidad de automatización adicional. Los únicos beneficios que puede reclamarse son los que corresponden al resultado directo de la automatización.

3. *Trabaje con los salarios de desarrolladores al valor real.* Hay dos realidades fundamentales de desarrollo de *software* que hacen que la estimación del costo de un proyecto de *software* sea único: 80% del costo de los salarios es del desarrollador y los salarios de desarrolladores crece a un ritmo mayor que la inflación. La implicación es que, aunque un programador junior gane hoy $60,000, por la misma posición dentro de un año tal vez se tenga que pagar $66,000, un aumento del 10%, aunque la tasa de inflación sólo sea del 2%. La implicación es que el valor actual neto (VAN) de un desarrollador en el futuro es mayor que el de hoy, algo que rara vez sucede en cualquier otra industria.

4. *Busque los beneficios primero.* Es probable que primero se deba analizar los beneficios de un proyecto de *software*, y luego determinar si se puede permitir el gasto en el área de TI.

2.4. Conceptos de calidad (CMM) en el análisis de sistemas

La siguiente tabla muestra la evolución de los conceptos de calidad desde 1930 hasta nuestros días.

Año	Nombre	Modelo / Método	Descripción
1930	Walter Shewhart	Control estadístico de la calidad. Propuso el conocido ciclo de calidad PDCA (plan-do-check-act)	Es considerado como el padre del control estadístico de calidad. Fue el primero que realizó investigaciones sistemáticas en materia de calidad y desarrolló métodos estadísticos, gráficos de control de calidad, aplicables al sector industrial. Sus

			investigaciones ayudaron a las empresas a disminuir el porcentaje de defectos y cumplir las especificaciones de los diseños.
1980	Phil Crosby	Cero defectos. “Parrilla de la madurez de la gestión de la calidad”. (Crosby 1980, 1990).	Remarcó la importancia de la implicación y de la motivación de cada uno de los empleados en la empresa. Destacó los factores que representan los costos de la calidad y de las no conformidades y propuso una tabla que especifica cinco niveles de madurez. Tiene cierto paralelismo con los cinco niveles de madurez del modelo CMM.
1982	Edwards Deming	Aplicó y extendió el uso del ciclo PDCA (Deming. 1982).	Deming fue el impulsor de uno de los principios básicos de calidad, el famoso ciclo PDCA, también conocido como rueda de Deming. Propuso 14 puntos esenciales en la gestión de una empresa para conseguir la transformación y mejora continua en la organización.
1988	Joseph Juran	*La trilogía de Juran*. (Juran 1990,1999).	Desarrolló y aplicó con éxito los principios de Shewhart. Propuso una aproximación sistemática para controlar y mejorar la

			calidad. Desplegó su famosa trilogía: la planificación de la calidad, el control de la calidad y la mejora de la calidad, que representan los instrumentos del directivo para administrar la gestión de calidad.
71989	Watts Humphrey	Creación de Modelo CMM. (Humphrey, 1989a).	A partir de la parrilla de madurez de Crosby, aplicó los conceptos de calidad a los procesos de desarrollo de *software* y propició la creación del modelo CMM.

Tabla 2.4.1. Movimientos globales de calidad habidos a lo largo de la historia.[17]

El modelo de madurez y de capacidad (CMM, *capability maturity model*) comprende los elementos básicos para la administración de procesos eficaces, y tiene su fundamento en los principios creados por Crosby, Deming, Juran y Humphrey.

Desarrollo del CMMI[18]

CMM Integration surge como un proyecto con la finalidad de solucionar el problema de usar múltiples modelos CMMs para ser implementados por las organizaciones que pretendan mejorar los procesos dentro de su institución.

[17]Tuya, J., Ramos, R. I. y Dolado Cosin J. *Técnicas cuantitativas para la gestión en la ingeniería del* software. Netbiblo. S. L. 2007.

[18] Security by Design with CMMI for Development, Version 1.3. Disponible en http://goo.gl/hfRKe3. Recuperado: 14/10/2013.

El desarrollo de un conjunto de modelos integrados implicó más que una simple combinación de los materiales de los modelos existentes. Al usar procesos que fomentan el consenso, el equipo del producto CMMI creó un marco que da cabida a múltiples constelaciones. El primer modelo a desarrollar fue el CMMI para Desarrollo (entonces denominado simplemente CMMI). Inicialmente, CMMI era un modelo que combinaba tres modelos fuente: Capability Maturity Model for Software (SW-CMM) v2.0 draft C, Systems Engineering Capability Model (SECM) (EIA 2002ª) e Integrated Product Development Capability Maturity Model (IPD-CMM) v0.98. Los tres modelos fueron seleccionados debido al éxito en su adopción o por su prometedor enfoque para mejorar los procesos en una organización.

El primer modelo CMMI (V1.02) fue diseñado para usarse por organizaciones de desarrollo en su búsqueda de la mejora de procesos para toda la empresa. Fue publicado en 2000. Actualmente, rige la versión 1.3 publicada en 2010.

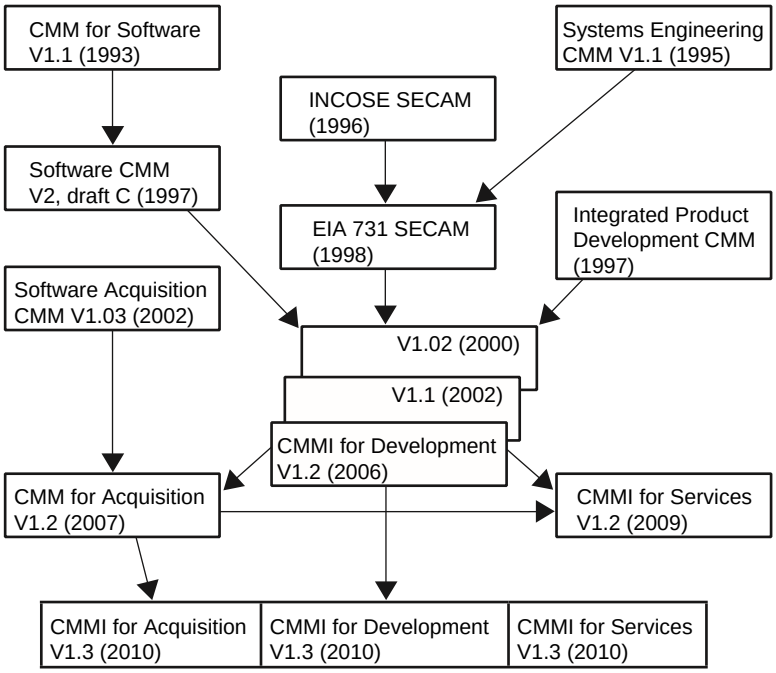
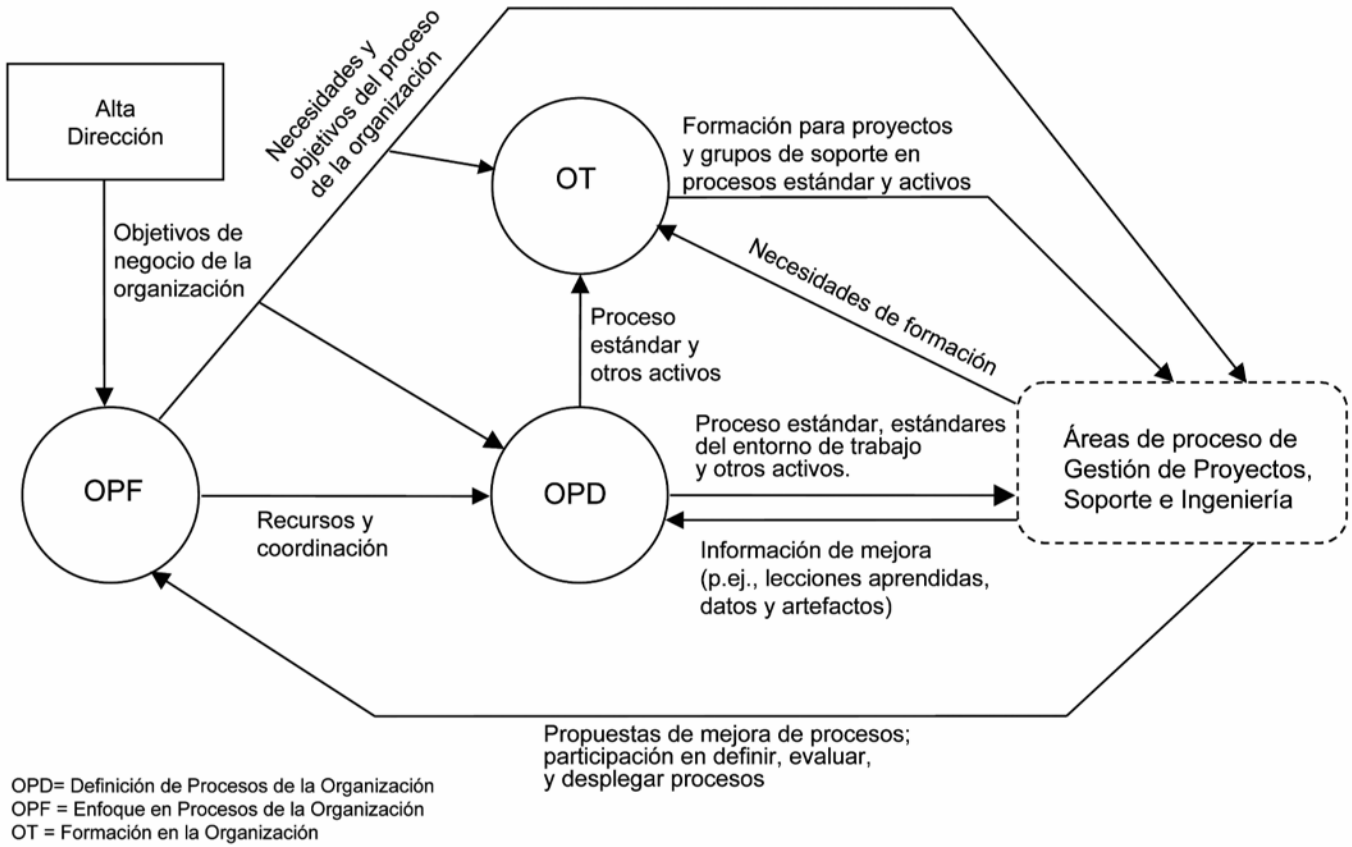


Figura 2.4.1. Historia de los CMMs.[19]

[19] En *Evolution of CMMI*. Disponible en http://goo.gl/FjOdtN. Recuperado: 08/10/2013.

CMMI para desarrollo

CMMI para desarrollo es un modelo de referencia usado tanto en productos como en servicios. Contiene prácticas que cubren la gestión de proyectos, la gestión de procesos, la ingeniería de sistemas, la ingeniería de *hardware*, la ingeniería de *software* y otros procesos de soporte utilizados en el desarrollo y mantenimiento.



Basado en CMMI-DEV 1.2. Disponible en: http://chrguibert.free.fr/cmmi12/cmmi-dev/text/7938cba-6019.php . Recuperado: 13/08/2014.

2.5. Análisis de requisitos del sistema

La fase de análisis de requisitos del sistema tiene como objetivo proporcionar una descripción completa de la cuestión sobre la base de los conceptos definidos en el ámbito del problema. Comienza con el análisis de los requisitos y se detiene cuando el modelo resultante se considera que está en conformidad con los requisitos señalados, o al llegar a un acuerdo entre los diseñadores y las partes interesadas.

Los requisitos funcionales y los requisitos no funcionales se identifican utilizando técnicas clásicas, como los casos de uso. Cada requisito se asocia entonces a una organización. Una organización se define por un conjunto de roles abstractos, sus interacciones y un contexto común, determinados de acuerdo con una ontología. En el proceso, primero se describe el contexto de la aplicación, a continuación se identifican las estructuras y finalmente se descomponen para interactuar arreglos más simples.

2.6. Especificación funcional del sistema

Trata con las descripciones ejecutables del sistema a desarrollar. La principal motivación es mostrar que los lenguajes de programación funcionales son útiles en la práctica de ingeniería de *software*.

Todo tipo de sistemas, ya sea con o sin un estado, se pueden modelar de manera simple. Se hace un intento de idear modelos ejecutables susceptibles de pruebas matemáticas. Además, es una técnica viable para el establecimiento de la corrección de algoritmos.

Como todo proceso deben imponerse limitantes y restricciones, particularmente en lo relacionado con la forma y tipo de lenguaje informáticos a usar. Esto trae como consecuencia, un lenguaje de especificación ejecutable.

2.6.1. Identificación y definición de requisitos

Dado que el *software* siempre es parte de un sistema, debemos iniciar por establecer los requisitos para todos los elementos y posteriormente generar un subconjunto de estas salvedades para el *software*. Esto es esencial cuando el *software* debe interactuar con otros elementos como el *hardware*, personas o bases de datos. De esta manera, se deben generar cuestionamientos que ayuden a comprender el dominio del flujo de información hacia el *software*, las funciones a realizar, el comportamiento y rendimiento, la forma de interfaz, etcétera.

Los requisitos tanto del sistema como del *software* deben ser documentados y revisados con el cliente.

2.6.2. Diseño del modelo lógico actual

Un modelo lógico es una herramienta de planificación para aclarar y representar gráficamente lo que el proyecto pretende hacer y lo que espera lograr, así como su impacto. Se distingue por lo siguiente:

- Resume los elementos clave del programa.
- Explica la razón de ser de las actividades del programa.
- Aclara los resultados esperados.
- Proporciona una herramienta de comunicación.

Debemos pensar en un modelo lógico como un mapa que se desarrolla para aclarar y comunicar las intenciones del proyecto de hacer y su posible impacto.

Los componentes de los modelos lógicos varían porque no existe un formato único de modelo lógico. De acuerdo con la Guía de Desarrollo de la Fundación WK Kellogg[20], los componentes de un modelo lógico son explicados en la siguiente tabla.

[20] W. K. Kellogg Foundation. *Logic Model Development Guide.* Disponible en http://goo.gl/yMPkLs. Recuperado: 16/10/2013.

RECURSOS/ INSUMOS	ACTIVIDADES	PRODUCTOS	RESULTADOS	IMPACTO/ META
Los recursos dedicados o consumidos por el programa. Incluyen el capital humano, recursos financieros, así como todo lo necesario para que pueda realizarse de forma exitosa el trabajo.	Lo que hace el programa con las entradas para cumplir su misión. Involucra los procesos, herramientas, eventos, tecnología y acciones que permiten originar los cambios o resultados intencionados del programa.	Los productos generados directamente de las actividades del programa. Pueden incluir tipos, niveles y metas de servicios que serán prestados por el programa.	Beneficios para los participantes durante y después de las actividades del programa. Son las consecuencias directas del comportamiento, conocimiento, aptitudes, condición y nivel de funcionamiento de los participantes del programa.	Resultados deseados del programa a largo plazo. Son las transformaciones intencionadas o involuntarias fundamentales que ocurren en las organizaciones, comunidades o sistemas como resultado de las actividades del programa dentro de 7-10 años.

A veces, un modelo lógico del programa se confunde con un plan de acción del proyecto. Si bien existe cierta superposición, la diferencia es sutil pero muy importante. En todo caso, después de haber definido los objetivos de los proyectos, productos y resultados, será relativamente fácil desarrollar un modelo de lógico del programa.

Cómo interpretar un modelo lógico

Debe ser considerado de izquierda a derecha. Los modelos lógicos muestran los aspectos básicos del programa en el tiempo, desde la planificación hasta los resultados. Comprender un modelo lógico significa seguir la cadena de razonamiento o enunciados "si…, entonces…" que conectan las partes del programa.

2.6.3. Estudio de alternativas de construcción

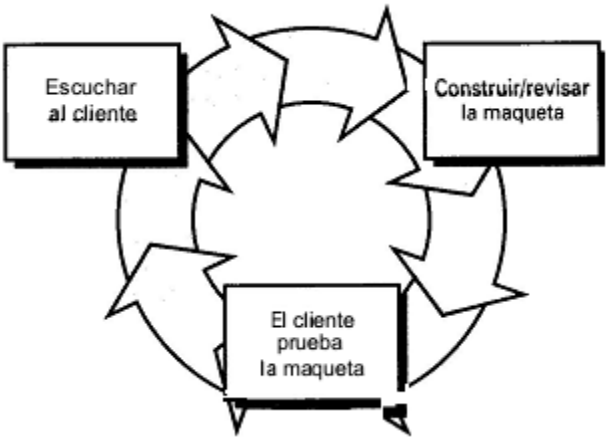


Figura 2.6.3. Paradigma de construcción de prototipos.

Al momento de planear las necesidades de un desarrollo de *software*, deben considerarse caminos alternos a su construcción. Esto implica en muchas ocasiones resolver el mismo problema, aunque desde perspectivas diferentes, sin dejar de lado el objetivo real por el cual fue concebida la idea. Este procedimiento debe ser documentado y señalado como "estudio de alternativas", y requiere la gestación de nuevos modelos de proceso según la naturaleza del proyecto, incluyendo métodos y herramientas a utilizarse, así como controles y entregas requeridos.

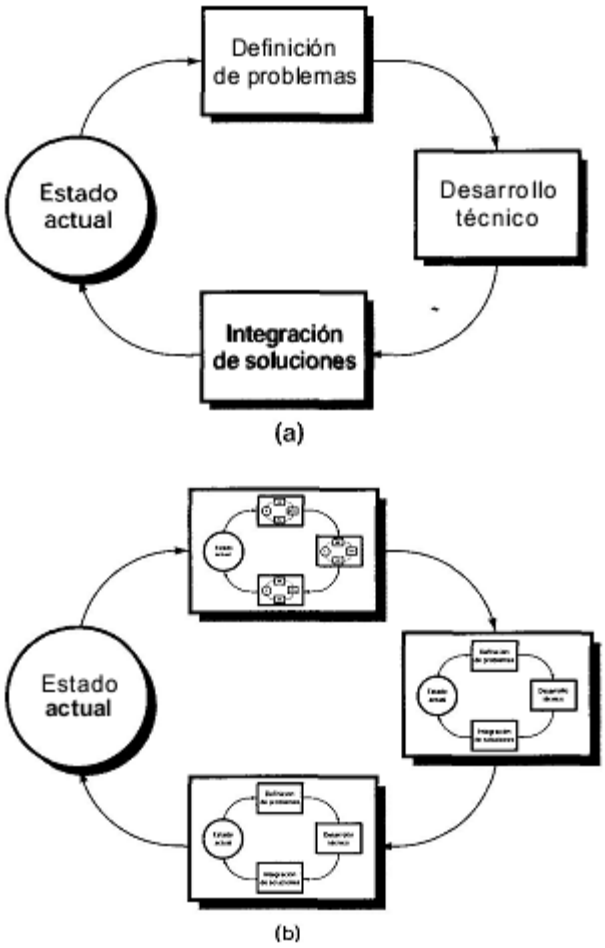


Figura 2.6.4. (a) Fases de un bucle de resolución de problemas.
(b) Fases dentro de las fases del bucle de resolución de problemas.[21]

La falta de esta información dificulta apreciar de forma razonable y concisa datos tan importantes como los costos, riesgo, designación de tareas y todas aquellas métricas que indiquen el estatus actual del proyecto.

[21] Pressman, R. *Ingeniería del* software. *Un enfoque práctico,* p. 19.

2.7. Especificación funcional del sistema

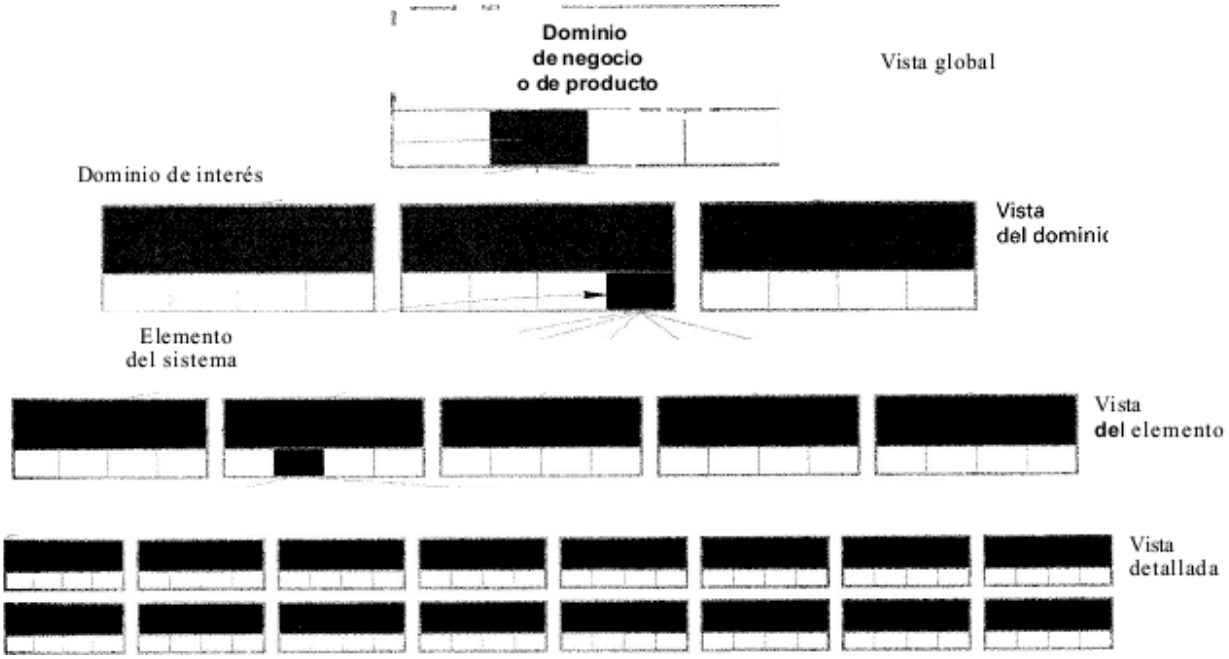


Figura 2.7. Evaluación del impacto.[22]

Es insuficiente contar con una implementación bien programada y estéticamente atractiva. El desarrollo debe ser funcional y cubrir todas las especificaciones establecidas. En esta línea, como ilustra la figura 2.7, el impacto de nuestro desarrollo de *software* comenzará desde una perspectiva global (lo que ve el usuario), hasta llegar a una vista detallada de cada etapa o proceso.

[22] Pressman, R. *Ingeniería del* software. *Un enfoque práctico,* p. 100.

2.7.1. Construcción del modelo de procesos del nuevo sistema

Los productos de *software* representan el resultado de un proceso de información intensiva, se construyen gradualmente y son revisados de manera intensa a través de un esfuerzo de desarrollo de *software*. Tales esfuerzos pueden ser organizados usando modelos del ciclo de vida del producto de *software*. Estos modelos de desarrollo de productos representan una evolución de los modelos de ciclo de vida del *software* tradicional (MacCormack, 2001).

Las revisiones se presentaron debido a la disponibilidad de nuevas tecnologías de desarrollo de *software*, como lenguajes de programación y entornos de creación de prototipos, *software* reutilizable, generadores de aplicaciones y entornos de apoyo de documentación. Cada una de estas tecnologías tiene como objeto permitir la creación de implementaciones de *software* ejecutables, ya sea de forma temprana dentro del esfuerzo de desarrollo de *software* o con mayor rapidez. En este sentido, los modelos de desarrollo de *software* pueden estar implícitos en el uso de la tecnología, ya que tales modelos se vuelven cada vez más intuitivos para los desarrolladores; y si a esto agregamos la experiencia con estas tecnologías, se justifica su empleo. Luego, la aplicación de un examen detallado de estos modelos es lo más apropiado cuando estas tecnologías están disponibles para el uso o la experimentación.

2.7.2. Construcción del modelo de datos del nuevo sistema

El modelado de datos es una forma de estructurar y organizar datos, y se aplica ampliamente en diversas industrias, no sólo en la generación de *software*, en razón de que cada vez se depende más de la gestión de bancos de datos.

Según el Instituto Americano de Estándares Nacionales (ANSI, por sus siglas en inglés), los modelos de datos pueden ser los siguientes:

- *Conceptual*. La idea a realizar (lo que pretendemos que sea).
- *Lógico*. Lo más razonable y viable de realizar.
- *Físico*. Lo que realmente será implementado o puesto en práctica.

En el mundo de los negocios, el modelado de datos resulta ser muy importante, sobre todo en la fase de diseño inicial. Además, durante el proceso, la comunicación y precisión son claves para que un modelo de datos sea crucial.

El proceso de análisis de *software* consiste en dos actividades principales: modelado de datos y modelado funcional, descritos a continuación.

Modelado de datos

El modelo de datos se crea como representación de las necesidades de información para generar un nuevo *software*. Funciona como herramienta de comunicación eficaz para conversaciones con los usuarios, y también sirve como un modelo para el sistema de base de datos. Así, actúa como un puente de información entre de los datos reales y el banco de datos donde serán almacenados los datos relevantes. Con esa información, se generan esquemas con los diferentes niveles de detalle a partir de los requisitos de más alto nivel.

Modelado funcional

El modelado funcional es el más utilizado a través de diagramas de flujo de datos (DFD) y metodologías orientadas a objetos (OO) que enfatizan el modelado de datos a través de diagramas de clases.

El enfoque ágil de modelado de datos usa UML, que incluye diversas técnicas, tanto para los datos, como en los modelos funcionales que pueden ser utilizados de varias maneras. De hecho, utilizar diferentes metodologías de modelado de datos y técnicas de modelado de proceso tiene ventajas y desventajas. Cabe aclarar que no hay ningún método estándar o modelo que nos permite modelar todos los aspectos de un almacén de datos (Mora y Trujillo, 2004).

El modelo de datos es una sección crucial del análisis de sistemas considerado el núcleo del almacenamiento de datos. Un modelado de datos es un plan para la construcción de una base de datos. Para ser eficaz, debe ser lo suficientemente simple para comunicarse con el usuario final. Además, se recomienda que la estructura de datos sea completamente detallada con la finalidad de generar la estructura física de la misma. Como fuere, determinar cuál de los tipos de modelos de datos funcionará debe basarse en el propósito del modelo y recursos disponibles.

2.7.3. Elaboración de análisis detallado del nuevo sistema

Se debe generar un documento que incluya todos los requisitos necesarios del nuevo sistema a desarrollar:

- *Aspectos técnicos (vista del desarrollador)*. Lista de los problemas técnicos relevantes para el proyecto actual.

- *Funciones básicas de la aplicación (vista del usuario)*. Se describen las características básicas principales para el proyecto actual de forma que sea entendible por el usuario final.

- *Casos de uso.* Son textos y diagramas que ilustran cómo están actuando todas las funciones de la aplicación y sub-características, y cómo se da la comunicación a través de varias etapas de uso y flujo de trabajo.

- *Especificación de interfaz de usuario*. Este documento es útil mientras se construye la aplicación. A través de esta especificación, el cliente tiene un primer contacto con el sistema y facilita la comprobación de lo que obtendrá al aplicarse todas las características requeridas.

- *Requisitos de usuario*. Descripción detallada de las características básicas que se han identificado.

- *Arquitectura*. Modelo de seguridad, configuración de aspectos, transferencia de datos entre aplicaciones, identificación de los módulos y sus límites.

- *Campos de datos*. Se establecen los campos a usar y el esquema de la base de datos, así como sus relaciones. Los datos son identificados y validados con el

cliente para garantizar que el análisis de las necesidades actuales cubra todos los requerimientos.

- *Estimación del tiempo.* De acuerdo con los módulos, pasos de desarrollo, el nivel de los expertos desarrolladores y los niveles lógicos y de ingeniería.

2.8. Interfaz del sistema

La interfaz es un límite compartido o conexión entre dos objetos diferentes, dispositivos o sistemas a través del cual se transmite la información. La conexión puede ser física o lógica. El término se utiliza en muchos campos. En la electrónica y equipo de ingeniería, una interfaz puede ser el límite físico entre dos subsistemas o dispositivos; una parte o circuito en algunos subsistema que envía o recibe señales hacia o desde otros sistemas o subsistemas (por ejemplo, una interfaz de video o una tarjeta de interfaz de red); o un estándar que especifica un conjunto de características funcionales, las características comunes de interconexión físicas y características de la señal para el intercambio de señales o datos (por ejemplo, SCSI o USB).

También puede ser una combinación de estas características, como un puerto serie o un puerto paralelo, que forma un límite físico. Se utiliza para la transmisión de señales entre los diferentes sistemas y se adhiere a estándares de la industria, tanto para características físicas y eléctricas.

Una interfaz de programación de aplicaciones (API) es un conjunto de especificaciones que definen cómo una pieza de *software* interactúa con otras, en particular un programa de aplicación en un sistema operativo. El objetivo principal es proporcionar un conjunto de funciones de uso común, como dibujar ventanas o iconos en la pantalla, lo que ahorra a los programadores la tediosa tarea de escribir código para todo desde cero.

El desarrollo y mejora de las interfaces puede ser una tarea difícil. Esto es particularmente cierto en lo que respecta a las interfaces de usuario, ya que para desarrollar interfaces de usuario verdaderamente de alta calidad, es indispensable tener en cuenta la complejidad de los temas de ergonomía y variabilidad humana (o sea, las diferencias en las características físicas y mentales entre los seres humanos).

2.8.1. Definición de interfaces con el usuario

Una interfaz de usuario es la forma en la que una persona controla una aplicación de *software* o un dispositivo de *hardware*. Una buena interfaz de usuario proporciona una experiencia “amigable" que permite al usuario interactuar con el *software* o el *hardware* de una manera natural e intuitiva. Casi todos los programas de *software* tienen una interfaz gráfica de usuario o GUI. Esto significa que el programa incluye controles gráficos que el usuario puede seleccionar, como el empleo de un ratón o teclado.

Una GUI típica de un programa de *software* incluye una barra de menús, barra de herramientas, ventanas, botones y otros controles. Los sistemas operativos Windows y Macintosh tienen diferentes interfaces de usuario, pero coinciden en muchos elementos, como una computadora de escritorio, ventanas, iconos, etcétera. Estos elementos comunes permiten que las personas puedan usar cualquier sistema operativo sin tener que volver a aprender por completo la interfaz. Del mismo modo, los programas como procesadores de texto y navegadores web tienen interfaces bastante similares, proporcionando una experiencia de usuario adecuada a través de múltiples programas.

2.8.2. Complementar las especificaciones del sistema

Una de las mejores maneras de asegurar la calidad de cualquier producto o servicio es conseguir los requisitos para hacerlo bien. Los desarrollos de *software* no son la excepción, las especificaciones pueden ser utilizadas para desarrollar una solución adecuada al problema, pero debemos ir más allá. Aparte de aplicar todo el proceso de obtención, análisis, comunicación y completar los requisitos para generar la base esencial en un desarrollo de *software*, es necesario tomar en cuenta cómo va a evolucionar a través del tiempo. Esto significa que tal vez los requisitos deben ser reutilizados en diferentes proyectos y a través de familias de productos para reducir el costo y esfuerzo de generar nuevo *software*.

2.8.3. Complementar las especificaciones de entrega

El propósito de un análisis de las necesidades y especificaciones es separar las necesidades de los deseos, y con ello evitar que el cliente pueda querer más de lo que está disponible. A menudo, la especificación está escrita en el idioma del cliente, quien firmará el documento. Aquí puede comenzar el problema: ¿cuántas personas de la organización hablan el idioma del cliente? Por lo general, pocos. Luego, debe haber un proceso que ayude a identificar las funciones de cada participante en el proyecto. Esto se hace durante el análisis de necesidades, ya que los requisitos pueden cambiar como resultado de los avances en el desarrollo del *software*.

Para cumplir y complementar las especificaciones de entrega, lo más recomendable es incluir a todos los miembros que participan dentro del proyecto de gestión de *software*: gestores de proyectos, analistas de negocio, programadores, evaluadores, gerentes de control de calidad, clientes, etcétera.

Este encuentro se traducirá en un pliego de condiciones, por lo que no habrá de generarse diseños o códigos hasta que estén concluidos el análisis de las necesidades y sus soluciones. Dado que la especificación está escrita en el idioma del cliente, hay necesidad de convertir el lenguaje del cliente en el idioma del grupo de desarrollo y el grupo de prueba. Por ejemplo, los programadores deben saber cómo hacer que la aplicación se ejecute en el sistema; o los probadores, entenderán cómo van a comprobar que la solicitud cumple con las necesidades del usuario (a los probadores no les importa el idioma en el que fue escrito el programa, sólo si cumple con las necesidades del usuario; deben conocer las expectativas de los usuarios y cómo los usuarios utilizan la aplicación). El objetivo general es, en todo caso, ofrecer un producto que satisfaga al cliente.

Una vez traducidas las especificaciones, se genera un tutorial. Muchos de los subgrupos de desarrollo hacen tutoriales sobre el diseño, código, etcétera. Con todo, los proyectos de *software* pueden fallar debido a la falta de comunicación sobre el significado de lo que dijo un cliente frente a lo que en realidad es; malentendidos que pueden llevar a suposiciones erróneas.

RESUMEN DE LA UNIDAD

Uno de las fases más importantes en la creación de sistemas de información es el análisis, en el cual será posible determinar si se cuenta con los recursos necesarios para realizar el sistema. El análisis debe partir del establecimiento de objetivos precisos acordes con los objetivos de la organización, luego, pasaremos a realizar los estudios de viabilidad que abarcan los aspectos técnico-financieros en donde se determina si se cuenta con los recursos materiales, financieros y humanos necesarios para la realización del proyecto.

Una vez que los estudios de viabilidad son realizados y el proyecto aprobado, se recomienda que los involucrados en su desarrollo sigan modelos de calidad, lo que significa emplear lo que se denomina "buenas prácticas de la ingeniería de *software*" para la creación del sistema. Al establecer el modelo de calidad a seguir, se iniciará con la fase de recopilación y análisis de requerimientos o requisitos, donde se recaba la información necesaria para describir la forma como trabajará el sistema y la manera de interactuar con su entorno. Por lo general, esta fase es una de las más complicadas, ya que en muchas ocasiones la falta de comunicación entre clientes y desarrolladores lleva a un mal entendimiento de los requisitos y, por ende, a un mal diseño del sistema.

Los requisitos recabados ayudarán a los ingenieros de *software* a detallar el modo funcional del sistema, identificar riesgos en su construcción, establecer alternativas para su elaboración y, finalmente, desarrollar interfaces visuales que ayuden a evaluar si lo que se desea construir va bien encaminado. La fase de análisis es crítica en el desarrollo de sistemas de información; si se realiza erróneamente, el sistema que resulte no cubrirá las expectativas y, en consecuencia, llegará a ser muy costoso para la organización.

GLOSARIO DE LA UNIDAD

Análisis de requisitos

Proceso de estudio de las necesidades del usuario para conseguir una definición de los requisitos del sistema o del *software*.

Desarrollador

Programador dedicado a una o más facetas del proceso de desarrollo de *software*; realiza programas o aplicaciones en uno o varios lenguajes de programación informática. Asimismo, realiza proyectos referidos a la ingeniería del *software*. Puede contribuir a la visión general del proyecto más a nivel de aplicación que a nivel de componentes o en las tareas de programación individuales.

Escenarios

Descripción de situaciones que se pueden presentar cuando los usuarios interactúan con los sistemas de información.

Ingeniería del *software*

Aplicación de procesos sistemáticos y disciplinados para el desarrollo, operación y mantenimiento de *software.*

Sistema

En sentido general, es un conjunto de cosas que se relacionan entre sí, ordenadamente, que contribuyen a determinado objeto; elementos interrelacionados y regidos por normas propias, de tal modo que pueden ser vistos y analizados como una totalidad. El sistema se organiza para producir determinados efectos, o cumplir una o varias funciones. En el contexto de la informática, designa un conjunto de *hardware* y *software* específicos.

Sistema de información

Conjunto de procedimientos ordenados que, al ser ejecutados, proporcionan información para apoyar la toma de decisiones y el control de la institución. No implica necesariamente el uso de computadoras; se puede acceder a la información utilizando un método mecánico o físico (por ejemplo, buscar un expediente en un archivador).

Sistema informático

Resulta de la interacción entre los componentes físicos, *hardware*, y lógicos, *software*. A éstos hay que agregarles el recurso humano, parte fundamental de un sistema informático. Una simple computadora es un sistema informático, dado que al menos dos componentes deben trabajar en conjunto.

ACTIVIDADES DE APRENDIZAJE

ACTIVIDAD 1

Elabora un mapa mental sobre los procesos a seguir en la fase de análisis de sistemas.

ACTIVIDAD 2

Investiga en tres fuentes distintas sobre los elementos que debe contener un estudio de viabilidad de un sistema de información. Elabora una síntesis de tu investigación en no más de dos páginas. Menciona tus fuentes de consulta.

ACTIVIDAD 3

Discute con tus compañeros sobre la calidad en los sistemas de información, con base en las siguientes preguntas:

1. ¿Por qué es necesario un modelo de calidad de sistemas?
2. ¿Cuál es la importancia de seguir un modelo de calidad en el desarrollo de sistemas?

ACTIVIDAD 4

Realiza una investigación sobre el modelado de procesos y datos en el desarrollo de sistemas de información. Con base en lo anterior, desarrolla tres ejemplos de modelado de procesos y de datos empleados en la ingeniería de sistemas. Menciona tus fuentes de consulta.

CUESTIONARIO DE REFORZAMIENTO

Responde las siguientes preguntas:

1. ¿Cómo deben ser planteados los objetivos de un sistema de información en la etapa de análisis?
2. ¿Qué es un estudio de viabilidad?
3. ¿Cuáles son los objetivos de un estudio técnico-financiero?
4. ¿Por qué es importante la realización del estudio técnico?
5. Menciona dos ejemplos de modelos de CMM.
6. Explica brevemente por qué es importante la definición de requisitos en la etapa de análisis de sistemas.
7. ¿Qué es un modelo lógico del sistema de información y para qué es útil?
8. ¿Por qué es necesario establecer diversas alternativas de construcción de un sistema en la etapa de análisis?
9. Menciona cinco ejemplos de riesgos que pueden presentarse en el desarrollo de sistemas.
10. ¿Qué es el modelado de procesos y datos y cuál es su utilidad?

LO QUE APRENDÍ

Retoma el cuadro sinóptico que desarrollaste en la actividad “Lo que sé”, y escribe una comparación sobre los conceptos que plasmaste en él y los conceptos contenidos en la presente unidad. Escribe tus conclusiones al respecto.

EXAMEN DE AUTOEVALUACIÓN

Responde verdadero (V) o falso (F).

1. El objetivo del estudio de viabilidad del sistema es el análisis de un conjunto concreto de necesidades para proponer una solución a largo plazo, que tenga en cuenta restricciones económicas, técnicas, legales y operativas. ( )
2. El análisis económico incluye el análisis costo-beneficio. ( )
3. Phil Crosby destacó “la importancia de la implicación y de la motivación de cada uno de los empleados en la empresa”. ( )
4. El modelo CMMI para adquisición fue publicado en 2005. ( )
5. La versión 1.3 de CMMI para adquisición fue publicado en 2010. ( )
6. CMMI para desarrollo es un modelo de referencia que cubre las actividades para desarrollar solamente productos. ( )
7. El análisis de los requisitos es una tarea de ingeniería del *software* que cubre el hueco entre la definición del *software* a nivel sistema y las pruebas del *software*. ( )
8. Un modelo lógico es una forma sistemática y visual de presentar y compartir su comprensión de las relaciones entre los recursos que dispone para operar su programa, las actividades que planea realizar y los cambios o resultados que espera obtener. ( )
9. El propósito de un modelo lógico es suministrar a los participantes un mapa guiado que describa la secuencia de eventos relacionados que conectan la necesidad de un programa planificado con los resultados deseados. ( )
10. Joseph Juran fue el impulsor de uno de los principios básicos de calidad, el ciclo PDCA. ( )

MESOGRAFÍA

BIBLIOGRAFÍA RECOMENDADA

Autor	Capítulo	Páginas
Peña Ayala	3	15-43
Roger S. Pressman. *Ingeniería del* software*. Un enfoque práctico*	10	165-180

BIBLIOGRAFÍA BÁSICA (REFERENCIAS)

Peña Ayala, Alejandro. *Ingeniería de* software: *una guía para crear sistemas de información*. Instituto Politécnico Nacional.

Pressman, Roger S. *Ingeniería del* software. *Un enfoque práctico,* 5.ª ed. España: McGraw-Hill, 2002, 601 pp.

BIBLIOGRAFÍA COMPLEMENTARIA (TEMARIO ANALÍTICO)

1. Bardou, Louis. *Mantenimiento y soporte logístico de los sistemas informáticos.* México: Alfa Omega-Marcombo, 2004, 292 pp.
2. Bochinno, William A. S*istemas de información para la administración, técnicas e instrumentos.* México: Trillas, 2002.
3. Bonsón, Enrique. *Tecnologías inteligentes para la gestión empresarial*. México: Alfa Omega-Rama, 2002, 258 pp.
4. Cornella, Alfons. *Información digital para la empresa, una introducción a los servicios de información electrónica.* México: Alfa Omega-Marcombo, 2004, 196 pp.

5. Lardent, Alberto R. *Sistemas de información para la gestión empresarial, procedimientos, seguridad y auditoría.* Buenos Aires: Pearson Education-Prentice Hall, 2001.
6. Levine, Guillermo. *Computación y programación moderna*. México: Addison Wesley, 2000.
7. Long, Nancy y Larry Long. *Introducción a las computadoras y a los sistemas de información*, *edición Internet.* México: Prentice Hall, 1999, 416 pp.
8. McLeod, Raymond, Jr. *Sistemas de información gerencial,* 7.ª ed. México: Prentice Hall, 1999, 688 pp.
9. Oz, Effy. *Administración de sistemas de información,* 2.ª ed. México: Thomson-Learning, 2001, 578 pp.
10. Peña R., Baeza-Yates, R. y Rodríguez, J. *Gestión digital de la información. de bits a bibliotecas digitales y la web.* México: Alfa Omega-Rama, 2004, 464 pp.
11. Piattini, Mario; J. Antonio Calvo-Manzano, Joaquín Cervera y Luis Fernández. *Análisis y diseño detallado de aplicaciones informáticas de gestión.* México: Alfa Omega-Rama, 2004, 728 pp.
12. Stair, Ralph M. *Principios de sistemas de información*, 4.ª ed. México: Thomson Learning, 2002, 692 pp.
13. Walker, D. W. *Sistemas e información para la administración.* México: Alfa Omega-Marcombo, 2001, 360 pp.

SITIOS ELECTRÓNICOS

Dirección electrónica	Descripción
http://www.sei.cmu.edu/	Sitio oficial del Instituto de Ingeniería de Software de la Universidad Carnegie Mellon.
http://goo.gl/93ZAm4	Análisis de sistemas de información, de Luis Antonio Domínguez, de Red Tercer Milenio.

UNIDAD 3
DISEÑO DE SISTEMAS

OBJETIVO ESPECÍFICO

Al terminar la unidad, el alumno reconocerá los elementos que integran la fase de diseño de un sistema de información y conocerá diversas metodologías empleadas en esta fase.

INTRODUCCIÓN

El ciclo de vida de los sistemas de información parte con el análisis, donde se recaba la información necesaria para poder construir el sistema y, en todo caso, determinar si su creación es posible a través de diversos estudios de viabilidad. Concluida esta fase, se pasa al diseño, donde se establecerá la estructura funcional del sistema a nivel lógico, lo que permitirá la construcción del sistema físico.

En la fase de diseño, se emplean herramientas y técnicas que permiten a los diseñadores construir la estructura lógica del sistema. Aquí, se determina cómo estará estructurado el sistema, cómo circulará la información a través de él, la forma de almacenamiento y la estructura de los datos almacenados y, si es el caso, los procesos para la toma de decisiones que serán incluidos en el sistema. En este momento, así como en la fase de análisis, es posible seguir una o varias metodologías que permiten realizar las actividades asociadas a esta fase de manera correcta; en la presente unidad serán abordadas varias de ellas.

LO QUE SÉ

Elabora una síntesis donde expongas los elementos que integran la fase de diseño de sistemas y los resultados obtenidos de ella.

TEMARIO DETALLADO (8 HORAS)

3.1. Fundamentos del diseño de sistemas

- 3.1.1. Diagramación de sistema
- 3.1.2. Diagrama de flujo de información
- 3.1.3. Diagrama del flujo de datos
- 3.1.4. Diccionario de datos
- 3.1.5. Tablas de decisiones
- 3.1.6. Árboles de decisión
- 3.1.7. PERT y CPM

3.2 Metodologías del diseño de sistemas

- 3.2.1. Análisis estructurado
- 3.2.2. Diseño estructurado
- 3.2.3. HIPO
- 3.2.4. ISAC
- 3.2.5. Warnier Orr
- 3.2.6. Jackson System Development (JSD)
- 3.2.7. *Bussines system planning* (BSP)
- 3.2.8. Otras

3.1. Fundamentos del diseño de sistemas

El diseño de sistemas basa gran parte de sus principios en el diseño de *software*, que a su vez implica la utilización de técnicas de diseño, codificación y prueba. Dentro del diseño, se desarrollan, analizan y documentan los arreglos de datos, la estructura del programa y las instrucciones del nuevo *software* o servicio.

Algunas técnicas de diseño:

- *Arquitectura de* software. La arquitectura de *software* incluye la estructura jerárquica de los módulos y la estructura de los datos.

- *Estructura del programa.* Se muestra la estructura de los módulos. Un diagrama de la estructura se utiliza para representar esto. Abarca también el número de módulos controlados directamente por otro módulo.

- *Estructuras de datos*. Las estructuras de datos representan la relación lógica entre los datos. Ejemplo: una matriz, vinculada listas, pila, cola, etcétera.

- *Modularidad*. Un *software* se puede dividir en elementos direccionables y separados (módulos), integrados para resolver el problema.

- *Abstracción*. Técnica en la que los detalles no deseados no se incluyen y sólo se otorga la información necesaria. En el nivel más alto de abstracción, la solución se afirma en términos generales. En los niveles más bajos de abstracción, la solución se da en detalle.

El diseño nos permite interpretar con exactitud los requisitos del cliente en un producto o sistema, y sirve como base para las posteriores fases de desarrollo.

3.1.1. Diagramación de sistema

El uso de diagramas de flujo o diagramas de cajas es una forma rápida y fácil de crear una representación gráfica de datos o procedimientos.

3.1.2. Diagrama de flujo de información

El diagrama de flujo consiste en una serie de figuras que indican cada paso de un proceso. Cada figura especifica una función determinada, por ejemplo, un rombo representa una condición lógica y las flechas indican el flujo de control.

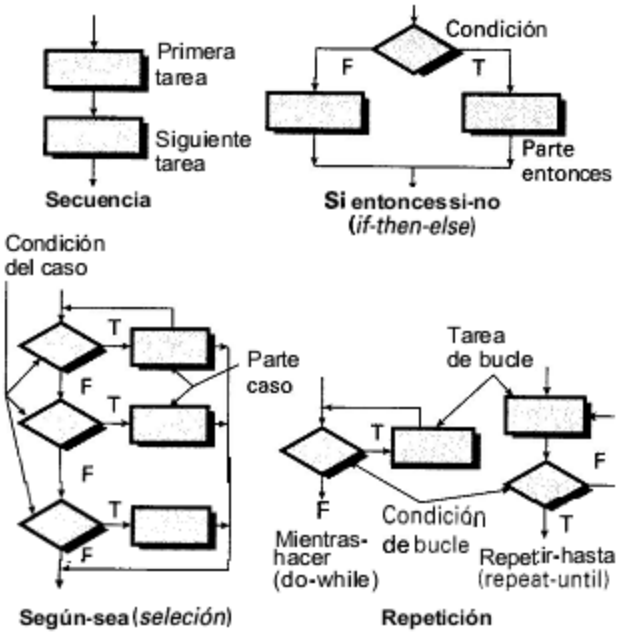


Figura 3.1. Construcciones en diagrama de flujo.[23]

[23]Paniagua, P. *Sistema de información para un verificentro de pruebas estáticas*. Disponible en http://goo.gl/qkNfT7. Recuperado: 12/08/2013.

La figura anterior (3.1) muestra tres construcciones estructuradas. Una secuencia se representa como dos recuadros de procesamiento conectadas por una línea (saeta) de control. La fabricación de selección de la figura realmente es una ampliación de si-entonces-si no. Un parámetro se prueba por decisiones sucesivas hasta que ocurre una condición verdadera y se ejecuta el camino de procesamiento asociado.

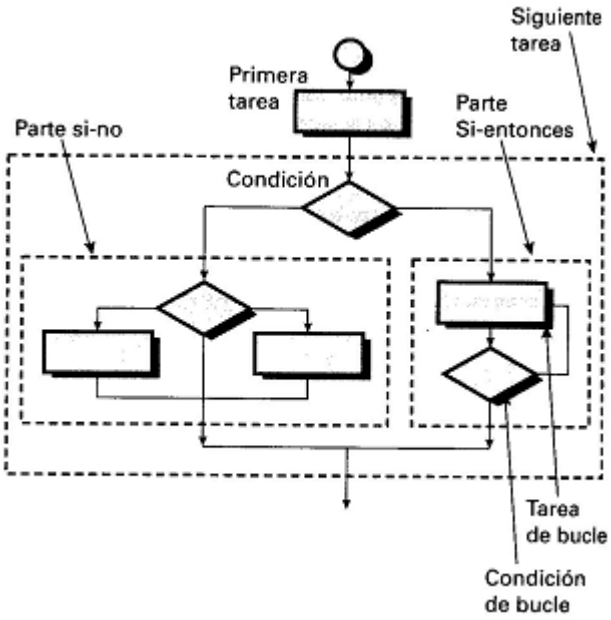


Figura 3.2. Construcciones anidadas.[24]

Las estructuradas pueden vincularse unas en otras como muestra la figura anterior (3.2). En esta figura, un repetir-hasta forma la parte then de un si-entonces-si-no (mostrada dentro de la línea discontinua exterior). Otro if-then-else forma la parte si-no de la primera condición. Por último, la condición propiamente dicha se convierte en un segundo bloque en una secuencia.

[24] Cedeño, Sarmiento y Herrera. *Sistema integrado para la automatización de un laboratorio clínico orientado a la web.* Disponible en http://goo.gl/4B2euC. Recuperado: 12/08/2013.

3.1.3. Diagrama del flujo de datos

El diagrama de flujo de datos (DFD) es un gráfico que indica el camino que deberán seguir los datos dentro de un sistema, por lo que podemos simplificarlo como el seguimiento del origen y destino de la información. Un diagrama de flujo de datos puede ser diseñado a partir de un nivel general hasta detallar cada uno de los procesos que integran el sistema. En el capítulo 8 de este material, en el documento que presenta el caso práctico, es posible revisar la forma de desarrollar un DFD a varios niveles de detalle. La siguiente figura ejemplifica un DFD general.[25]

[25] Manzano y Montesano. *Sistema Integral de Laboratorio Clínico Sofilab.* Tesis. Facultad de Ingeniería, UNAM. México, 2002.

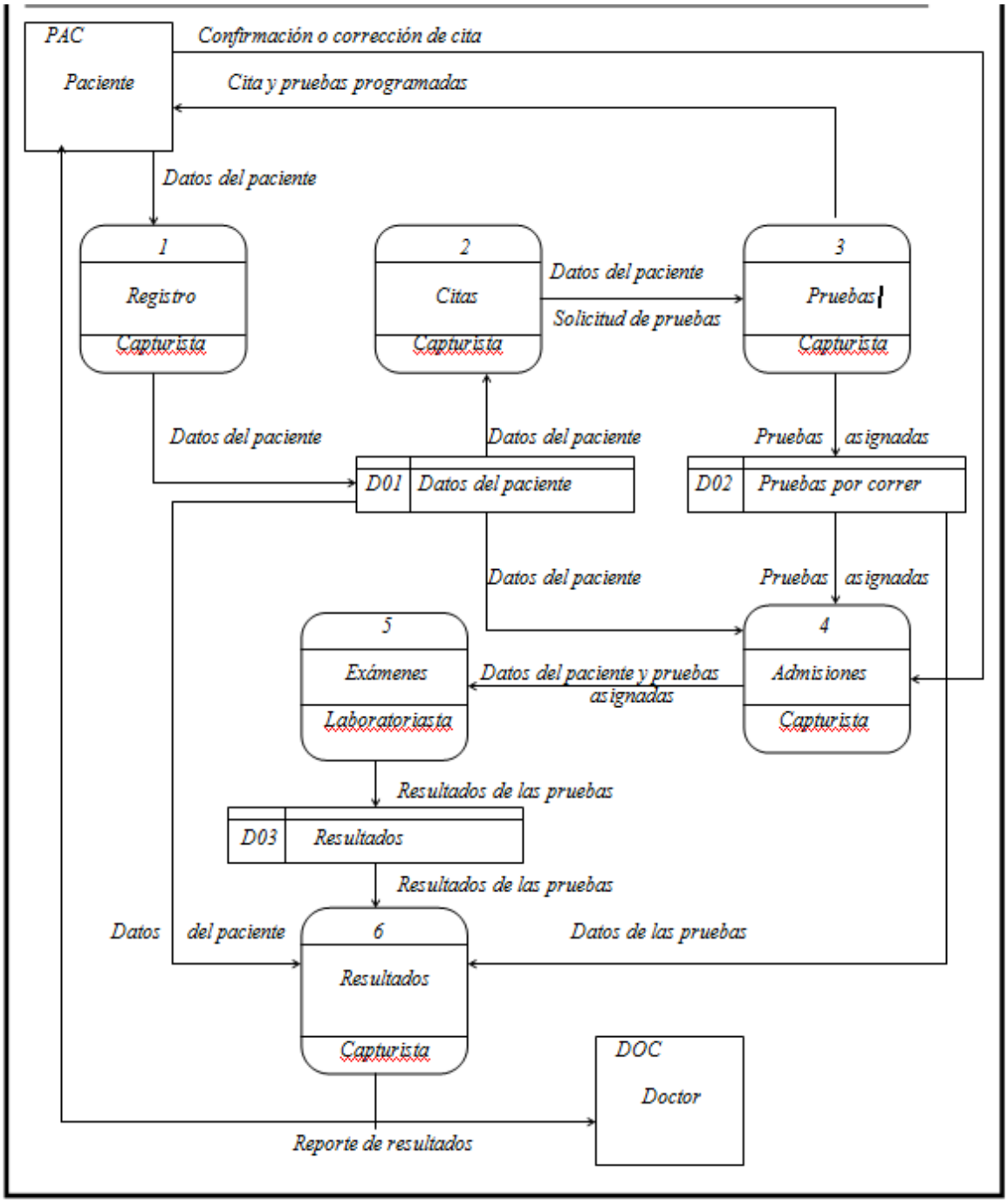


Figura 18.
Diagrama de flujo de datos general.

3.1.4. Diccionario de datos

El diccionario de datos consiste en una lista que incluye aquellos elementos de datos necesarios para que el sistema pueda ser interpretado, tanto por el usuario como por el analista del sistema. De forma general, muestra la estructura final de las tablas que constituirán la base de datos del sistema; y su construcción depende en gran medida del análisis resultante del diagrama entidad-relación.

Este diccionario incluye los siguientes elementos[26]:

- *Nombre.* El nombre cardinal del elemento de datos.
- *Alias.* Otros nombres empleados para definir el nombre.
- *Dónde se usa / cómo se usa.* Un listado de los procesos que usan el elemento de datos o de control y cómo lo usan.
- *Descripción del contenido.* Contenido representado mediante una notación.
- *Información adicional.* Otra información acerca de los tipos de datos, valores implícitos, limitaciones o restricciones.

A continuación se muestra un ejemplo de un diccionario de datos tomado del caso que se revisará en la unidad 8.

Tabla de valores de referencia. Esta tabla hace alusión a las claves de los diversos perfiles o pruebas a realizarse.

[26] Diccionario de datos. Disponible en http://goo.gl/usd6A7 y http://goo.gl/Ir3f7s. Recuperado: 12/08/2013.

CAMPO	LLAVE	NOMBRE	ABREVIATURA	TIPO	LONGITUD	NULO
1	Si	Clave de sección	Cve_Sección	Char	3	N
2	Si	Clave del perfil	Cve_Perfil	Char	3	N
3	Si	Clave de la prueba	Cve_Prueba	Char	3	N
4	Si	Clave de la fase	Cve_Fase	smallint	variable	N
5		Sexo del paciente	Sexo	Char	1	S
6		Fase de la prueba	Fase	Char	1	N
7		Rango de valores de la prueba	Rango	Varchar	25	S
8		Tipo de rango de la prueba	Tipo_Rango	Varchar	25	S
9		Descripción del tipo de rango	Descripción	Varchar	75	S
10		Valor cuantitativo 1	Vcuant1	Varchar	10	S
11		Valor cuantitativo 2	Vcuant2	Varchar	10	S

Tabla de usuarios. Guarda las claves de acceso, contraseñas (*passwords*) y niveles de acceso de cada usuario.

CAMPO	LLAVE	NOMBRE	ABREVIATURA	TIPO	LONGITUD	NULO
1	Si	Clave de usuario	Clave	Char	8	N
2		Contraseña del usuario	Pass	Varchar	8	N
3		Nombre del usuario	Nombre	Varchar	50	N
4		Nivel de acceso	Nivel	Char	3	N

Reactivos. Guarda los datos relacionados a los tipos de reactivos empleados para cada prueba, el número necesario en cada prueba y su cantidad en existencia.

CAMPO	LLAVE	NOMBRE	ABREVIATURA	TIPO	LONGITUD	NULO
1	Si	Clave de reactivo	Cve_reactivo	Char	2	N
2		Descripción del reactivo	Descripción	Varchar	50	S
3		Unidad del sistema MKS del reactivo	ReaUnidad	Char	10	S
4		Cantidad requerida por prueba	ReaCanPru	Char	4	S
5		Cantidad en existencia	ReaCanExi	Char	4	S
6		Cantidad máxima requerida	ReaCanMax	Char	4	S
7		Cantidad mínima requerida	ReaCanMin	Char	4	S

3.1.5. Tablas de decisiones

CONDICIONES		1	2	3	4
¿Paga contado?		S	S	N	N
¿Compra > $ 50000?		S	N	S	N
ACCIONES					
Calcular descuento 5% s/importe compra		X	X		
Calcular bonificación 7% s/importe compra		X		X	
Calcular importe neto de la factura		X	X	X	X

Figura 3.3. Tabla de decisión.[27]

[27] Castilla, M. *Sistemas de información*. Disponible en http://goo.gl/Ll02Gm. Recuperado: 12/08/2013.

La tabla de decisión, como su nombre lo indica, es una combinación de renglones y columnas que muestran condiciones y acciones. Desarrollado en la década de 1950 por la compañía General Electric, es empleada para diversas funciones, desde análisis de ventas, inventarios, control de rutas, hasta programación. Se conoce también como tabla de causa-efecto. Esto a razón de que existe una técnica de diagramación lógica asociada llamada gráfica causa-efecto, que a veces se emplea para ayudar a obtener la tabla de decisión, lo que describe Myers como una red de combinación lógica (Myers, 1979). Sin embargo, la mayoría de las personas les resulta más útil sólo la tabla descrita por Copeland en 2003.

Características

Las tablas de decisión proporcionan una forma sistemática de expresar estructuras complejas, lo cual es útil para los desarrolladores como para los evaluadores. Se pueden aprovechar en el diseño de prueba, sin importar si son usadas en las especificaciones, ya que ayudan a los probadores a explorar los efectos de las combinaciones de los diferentes insumos y otros estados en el desarrollo del *software*. Además ayudan a los desarrolladores a hacer un mejor trabajo, por ejemplo, al evaluar combinaciones, lo cual puede ser un reto, ya que el número de combinaciones a menudo puede ser enorme, y tratar de realizar todas las combinaciones puede ser poco práctico y hasta imposible. Por ello, es recomendable sólo realizar un pequeño subconjunto de combinaciones, seleccionando qué combinaciones debemos probar y cuáles dejar fuera.

¿Cómo usar las tablas de decisión para las pruebas de diseño de *software*?

La primera tarea es identificar una función o subsistema adecuado que reaccione de acuerdo con una combinación de entradas o eventos. El sistema no debe contener

demasiadas entradas, de lo contrario, el número de combinaciones se volverá inmanejable. La mejor forma de hacer frente a un gran número de condiciones es mediante la división en subconjuntos y resolver cada uno de ellos de forma individual. Una vez identificados los aspectos que deben ser combinados, se procederá a ponerlos en una tabla con todas las combinaciones de Verdadero y Falso para cada uno de los aspectos.

3.1.6. Árboles de decisión

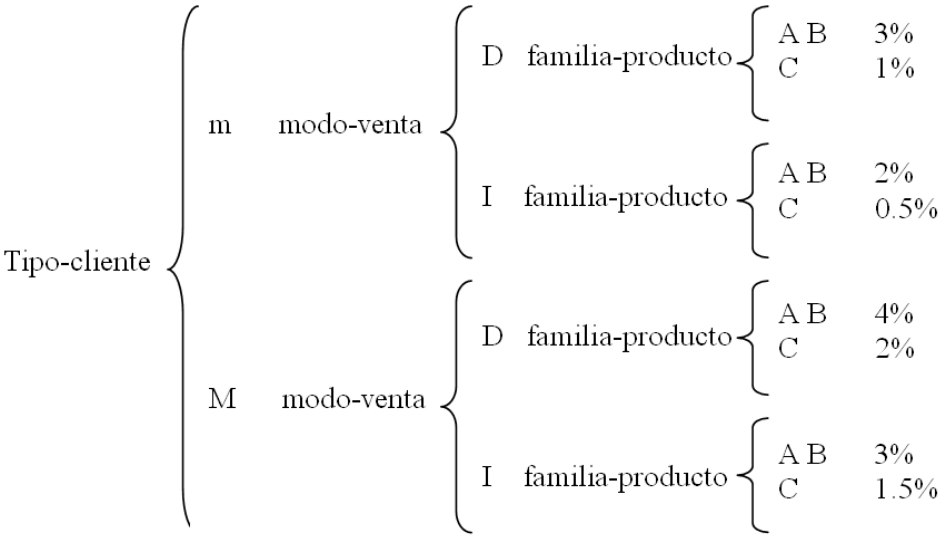


Figura 3.4. Árbol de decisión.

Definición

Es una técnica que facilita el análisis de decisiones progresivas con base en la aplicación de resultados y la asociación de sus probabilidades.

Beneficios

- Facilitan la elucidación de la decisión adoptada.
- Otorgan un mayor nivel de comprensión del conocimiento utilizado en la toma de decisiones.
- Revelan el comportamiento respecto a una determinada tarea de decisión.
- Aminoran el número de variables independientes.
- Funcionan como herramientas para el control de la gestión empresarial.
- Esquematizan los prototipos de inicio.
- Sus principales aplicaciones se dan en los procesos de búsqueda binaria, sistemas expertos y árboles de juego.

3.1.7. PERT y CPM

La técnica de evaluación y revisión de programa (PERT, *project evaluation and review technique*) y el método de la ruta crítica o del camino crítico (CPM, *critical path method*) están basados en redes que tienen como principal objeto asistir en la planeación, programación y control de proyectos. PERT es una herramienta probabilística y CPM es determinista.

Diferencias entre PERT y CPM

Lo que determina si se debe usar PERT o CPM es la forma como se realizan los cálculos de tiempo. Para el PERT, el tiempo para efectuar cada una de las actividades es una variable aleatoria descrita por una distribución de probabilidad; es un método basado en una red diseñada para ayudar en la planificación, programación, asignación de recursos y control de los proyectos; y ayuda en la planificación de la gestión adecuada de los proyectos complejos, el control de los planes y para mantener el plan al día. En cuanto a la técnica de estimación CPM, es un procedimiento que utiliza el análisis de redes para identificar las tareas que están en la ruta crítica; y ofrece una representación gráfica (visual) de las actividades del proyecto y el seguimiento de las actividades críticas.

PERT y CPM son también una herramienta útil para evaluar el rendimiento de las personas y equipos. En el ambiente de trabajo, las actividades están dispuestas primeramente en un orden secuencial y sus respectivos requisitos de tiempo se finalizan con base en la naturaleza del trabajo.

PERT también es empleado cuando las actividades están sujetas a una gran cantidad de variación (esto último es el resultado de muchos factores incontrolables, los cuales pueden ser observados y estimados, pero nunca dominados). Medimos su impacto por el aumento de la cuantía de la fuerza de trabajo que resulta en la disminución de tiempo de procesamiento para las actividades seleccionadas. La nivelación de recursos se logra al retrasar o adelantar el inicio de ciertas actividades, y ampliar la duración de determinadas actividades y, por tanto, la reducción de la demanda de recursos sobre la duración de la actividad o por una combinación de ambos de estos ajustes.

Estos procesos son muy utilizados para llegar a valores cuantitativos lógicos que nos ayudan en la gestión de las actividades de una manera científica. Así, CPM y PERT son herramientas de gran alcance y ayudan a programar o gestionar proyectos complejos.

3.2. Metodologías del diseño de sistemas

3.2.1. Análisis estructurado

El análisis estructurado comenzó a ser usado a finales de la década de 1960 y principios de 1970. Consiste en una técnica de ingeniería de *software* que utiliza diagramas y gráficos para representar y desarrollar las especificaciones del sistema que sean fácilmente entendibles por los usuarios. Estos diagramas describen los pasos a seguir y los datos necesarios para cumplir la función de un diseño de *software* en particular. Este tipo de análisis se centra principalmente en los sistemas y funciones lógicas, y su objetivo es convertir los requerimientos del negocio en programas de computadora y especificaciones de *hardware*.

Pasos a seguir más importantes en el proceso de análisis estructurado:

- Estudiar el entorno empresarial actual.
- Modelar el viejo sistema lógico.
- Modelar el nuevo sistema lógico.
- Modelar el nuevo entorno físico.
- Evaluar alternativas.
- Seleccionar el mejor diseño.
- Crear especificaciones estructuradas.

Existen tres vistas ortogonales relacionadas con el análisis estructurado:

- *Vista funcional*. Diagramas de flujo de datos que definen el trabajo realizado y el flujo de datos entre las cosas, proporcionando de esta manera la estructura primaria de la solución.

- *Vista de datos*. Comprende el diagrama entidad-relación y se enfoca en lo que existe fuera del sistema que se está supervisando.

- *Vista dinámica*. Incluye diagramas de transición de estado y define cuando las cosas suceden y las condiciones en las que pueden ocurrir.

3.2.2. Diseño estructurado

El diseño estructurado es una metodología sistemática para determinar las especificaciones de diseño de *software*, y presenta las siguientes características:

- Permite que el planteamiento del problema se oriente hacia la solución.
- Se basa en el principio de la simplificación de un sistema grande y complejo mediante la partición en módulos más pequeños.
- Favorece el uso de herramientas de gráficos para ayudar al diseño del sistema.
- Ofrece un conjunto de estrategias para el desarrollo de una solución.
- Ofrece un conjunto de criterios para la evaluación de un buen diseño.

3.2.3. HIPO

HIPO (*hierarchy-input-process-output*) fue desarrollado por IBM como una forma de representación para un desarrollo jerárquico de arriba abajo. Comprende una tabla de contenido, un conjunto de diagramas generales y un conjunto de diagramas a detalle. Aprovechado para la comunicación de las especificaciones del sistema, se compone de dos tipos de diagramas:

- Tabla de contenido visual (VTOC, *visual table of contents*). Muestra un arreglo de todos los módulos en una estructura jerárquica.

- Entrada proceso salida (IPO, *input process output*). Gráfica consistente en tres columnas, en donde se especifica cómo serán introducidos los datos, qué procesos serán realizados con ellos; y finalmente se muestran los resultados generados.

3.2.4. ISAC (*information systems work & analysis of change*)

ISAC es un método de origen sueco que tiene la intención de acompañar el desarrollo de sistemas de información desde el principio hasta el final. En realidad, el sistema rara vez se ocupa con este objetivo; pero los diagramas son herramientas prácticas y se utilizan en todo el mundo.

En ISAC, es posible aplicar diversos tipos de diagramas, aunque los más frecuentes son los de actividad A, excelente ayuda para dar una imagen clara de los flujos de información y las necesidades de información. Al haber reglas muy estrictas para el diseño de diagramas A, no pueden expandirse infinitamente como los diagramas de flujo estándar. Un diagrama A nunca es más grande que una página, pues hay que

definir un límite del sistema antes de empezar a diseñarlo. En este caso, si se requiere de más detalles, se diseña un "sub diagrama", de nuevo siguiendo reglas muy estrictas, centrándose en una de las actividades representadas en el primer diagrama.

Sistema

El total de las actividades a estudiar se llama *sistema*, representado como un cuadrado o un rectángulo vacío; y el contorno de este recuadro se denomina *frontera del sistema*, como lo muestra la siguiente figura.

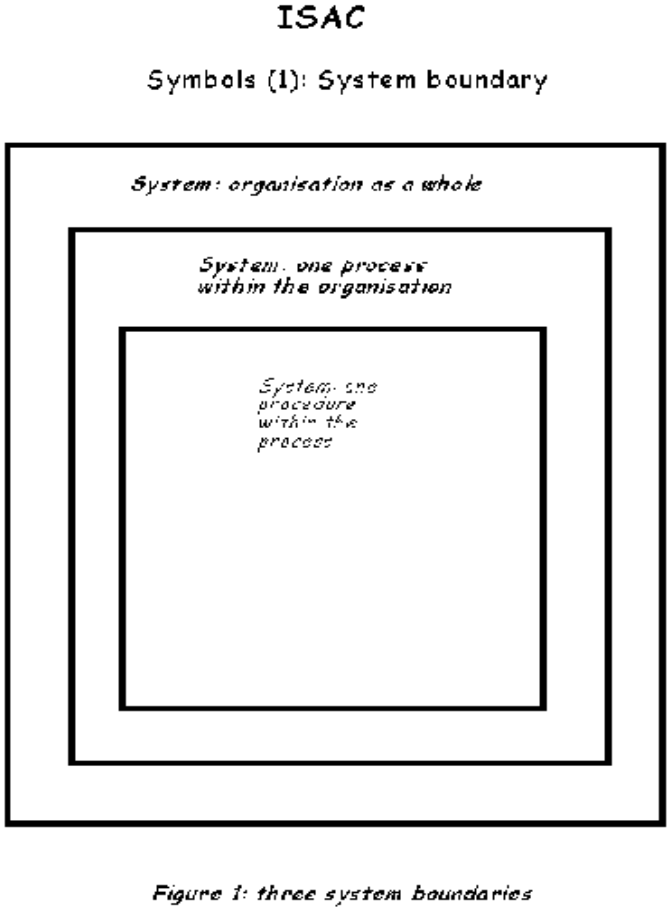


Figura 3.5 Diseño de un sistema usando ISAC[28]

[28] ISAC: Information Systems work & Analysis of Change. Disponible en http://goo.gl/0A7kgh. Recuperado: 28/10/2013.

El sistema a ser estudiado puede ser una organización en su conjunto. Pero si, por ejemplo, sólo está interesado en el proceso de compra, éste es el sistema; y todas las demás actividades de la organización están más allá de los límites del sistema, y así sucesivamente.

Entrada y salida

El sistema dispone de entrada y salida, y lo mismo se aplica para las actividades. Una actividad conduce a uno o más resultados, un producto. La actividad de "cálculo de los sueldos", por ejemplo, produce una nómina. Esta nómina es la salida o salida de conjunto de esta actividad. Pero antes de que las actividades generen un resultado, se han de cumplir algunas condiciones; son necesarios materiales o información para iniciar la actividad. Y estas necesidades se denominan conjunto de entrada o entrada.

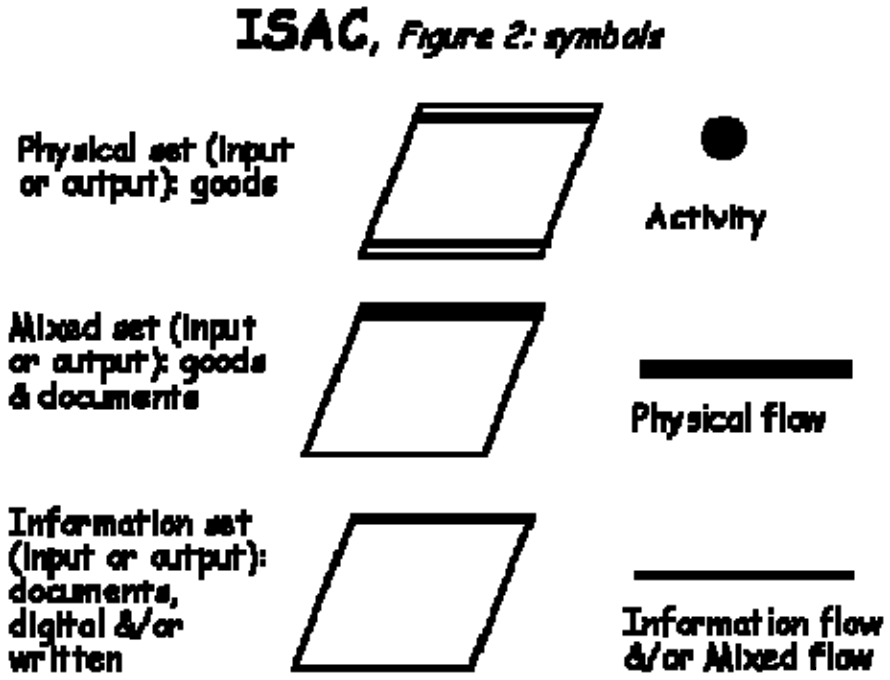


Figura 3.6 Símbolos usados en ISAC[29]

[29]ISAC: Information Systems work & Analysis of Change. Disponible en http://goo.gl/vOPHNc

Encabezado del diagrama

Cada diagrama tiene un encabezado como se muestra a continuación.

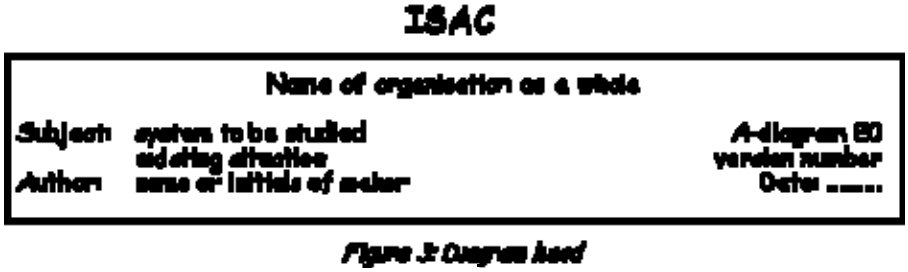


Figura 3.7. Encabezado en ISAC.[30]

Recuperado: 28/10/2013.

[30]ISAC: Information Systems work & Analysis of Change. Disponible en http://goo.gl/vOPHNc. Recuperado: 28/10/2013.

3.2.5. Warnier Orr

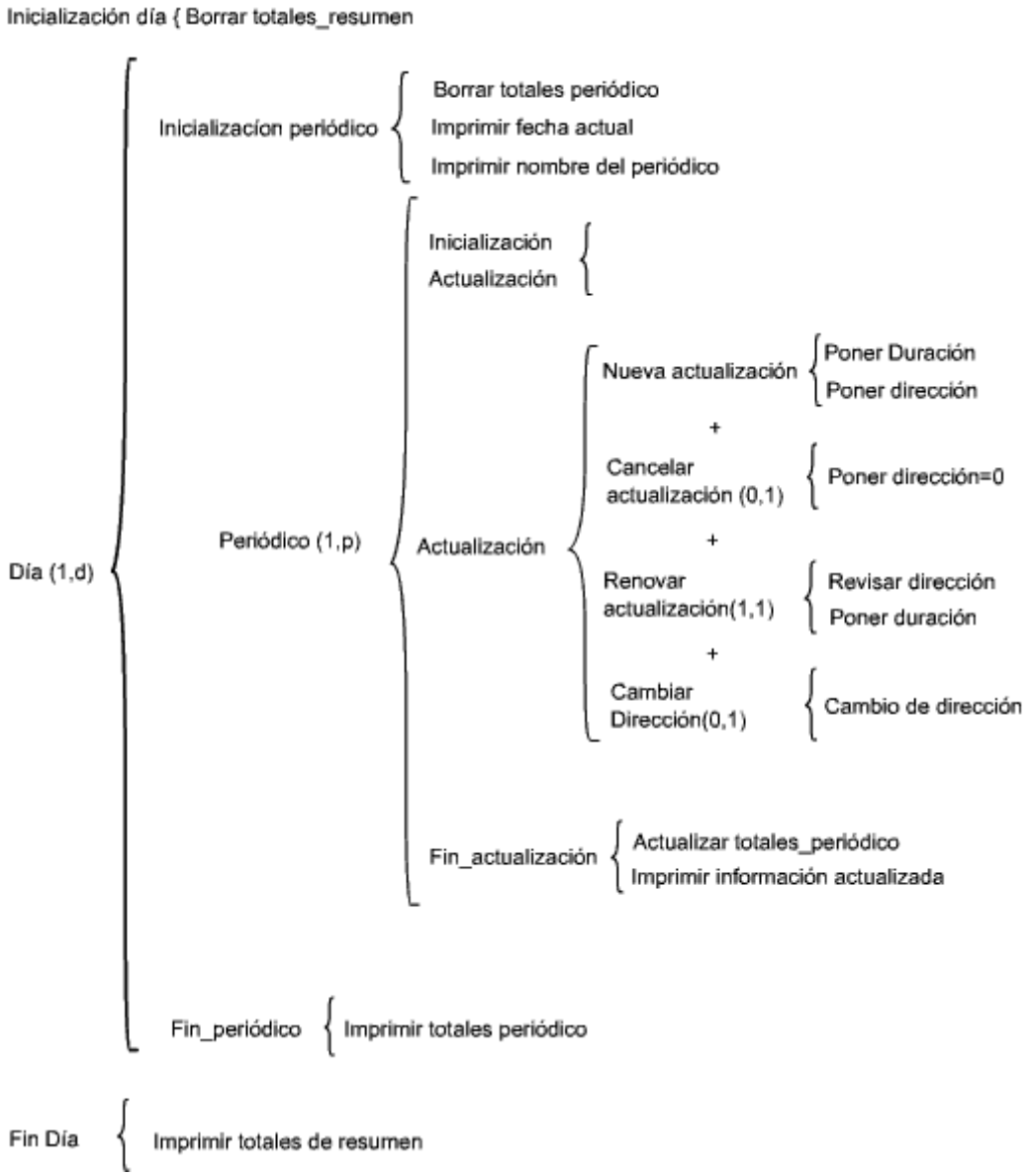


Figura 3.8. Ejemplo de diagrama Warnier/Orr.

Los diagramas Warnier-Orr se caracterizan por su claridad y simplicidad, y tienen como objetivo ser entendibles para todo el mundo. Esto significa que los diseños se crean para ser claros, rápidos y comprensibles. Su rango es tan amplio que puede abarcar desde listas "de lo que hay que hacer", hasta especificaciones de un programa

mediante una rigurosa metodología de estructura de datos. Son, pues, una herramienta conceptualmente simple, concisa y completa para visualizar información.

Organización

Un diagrama de Warnier-Orr es una jerarquía. Fluye de lo general a lo específico, un nivel a la vez. La raíz o concepto más general es la declaración que se ubica más a la izquierda. El diagrama comienza a la izquierda y procede a la derecha, y sólo se permite describir algo en un nivel a la vez.

Este diagrama se compone de dos elementos:

- *Declaraciones*. Simplemente declaran lo que se va a diagramar.
- *Extensiones de la declaración*. Es un nivel de desagregación que muestra en qué consiste la acción a realizar.

3.2.6. Jackson System Development (JSD)

Michael Jackson (no el cantante) creó el sistema de desarrollo Jackson (JSD), utilizando los principios establecidos en la programación estructurada Jackson (JSP). JSD es un método de análisis estructurado y de diseño similar a SSADM; emplea diagramas de estructura de entidad (ESD) y diagramas de red (ND) para modelar un sistema. Hoy, comprende sólo tres pasos:

- *Etapa de modelado (análisis)*. Se indican la acción y la entidad de las estructuras.
- *Fase de red (diseño).* Se indican el paso inicial del modelo, función y temporización del sistema.
- *Etapa de ejecución (realización).* Etapa de implementación.

3.2.7. *Business system planning* (BSP)

En la década de 1970, International Business Machines (IBM) inició el concepto de *business system planning* (BSP) o sistemas de planificación de negocios. La empresa comenzó con la idea de que todos los empleados de la compañía (técnicos o no) entendieran los conceptos de datos relacionales. Luego, dio origen a la técnica para que ellos gestionaran los datos almacenados. IBM desarrolló el proceso para uso interno, pero se popularizó entre sus clientes que en poco tiempo adoptaron esta metodología.

BSP se relaciona con el crecimiento de los negocios, que armoniza con la planificación estratégica. Utiliza la tecnología para llevar a cabo procesos de negocio que mejoren la organización, reduzcan los costos, faciliten la relación con las áreas funcionales y entreguen aplicaciones de forma eficiente a la organización que la implementa. Consiste en un proceso de siete pasos enfocados a mejorar cada aspecto de las áreas funcionales de la organización.

Paso 1. Aplicación

La implementación del proceso insta a los profesionales de negocios a la ejecución, a fin de llevar a cabo las tareas necesarias para ejecutar un plan de sistema de negocio exitoso.

Paso 2. Planear de forma rápida

El tiempo lo es todo, una planificación rápida posiciona a la organización a centrarse en no ser tan exacta.

Paso 3. Demostrar el valor de negocio en el plan

El análisis DAFO consiste en demostrar el valor de negocio en el plan. El concepto general es analizar dentro de las organizaciones las fortalezas y debilidades internas, así como las oportunidades y amenazas que se deben contemplar para facilitar el mejor plan de sistema de negocio y así lograr el objetivo de la organización.

Paso 4. Trabajo en equipo

El proceso de planificación de los sistemas de negocio debe ilustrar el trabajo en equipo. Trabajar en equipo insta a las organizaciones a entender la tecnología que se correlaciona con la planificación de los sistemas de información. Por tanto, las áreas funcionales pueden diseñar el sistema de negocio más acorde a los intereses de la organización.

Paso 5. Modelado

Modelar los datos refleja lo mencionado en el paso 2, el tiempo es todo. El modelado de datos en algunos casos puede ser lento y costoso.

Paso 6. Metodología

La micro gestión del sistema es beneficiosa para un BSP de éxito, para lo cual es idóneo la comprensión de herramientas informáticas y métodos para gestionar las funciones del sistema.

Paso 7. Consultoría

La consultoría debe manejarse con cuidado. La mayoría de las organizaciones prefieren consultores a aplicar un activo valioso. A veces la consultoría no es siempre la fórmula para un plan de negocio de éxito. El papel de los consultores debe ser un mecanismo de enseñanza para todos los individuos implicados dentro de la organización. La consultoría dentro de una organización, posiblemente, representa la falta de confianza en los sistemas de información o el departamento de tecnología de la información para ejecutar sus ideas originales.

3.2.8. Otras

NASSI-SHNEIDERMAN[31]

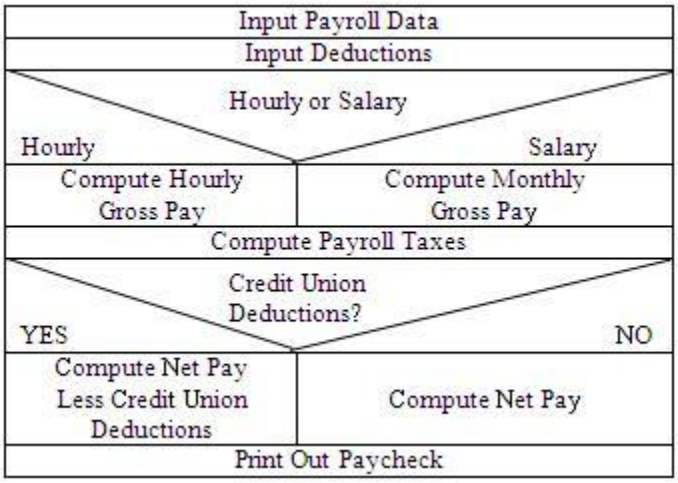


Figura 3.9. Diagrama NASSI-SHNEIDERMAN.

Desarrollado por Isaac Nassi y Ben Shneiderman, este diagrama es una forma de gráfico de programación que se asemeja mucho a las estructuras de control utilizados por los programadores; y describe el flujo de datos a través de un programa de computadora. Estos gráficos, también conocidos como diagramas de flujo

[31] System Analysis and Desing. Disponible en http://goo.gl/koxppf. Recuperado: 12/08/2013.

estructurados, son empleados por analistas y programadores porque muestran claramente la lógica de un programa.

RESUMEN DE LA UNIDAD

El desarrollo en el ciclo de vida de los sistemas de información es una etapa de consolidación de los resultados obtenidos en la fase de análisis. Aquí, se conceptualiza el sistema de manera funcional y se establece una metodología de diseño que se puede auxiliar de diversas herramientas para concretar el diseño conceptual del sistema. En esta fase, podemos emplear diversas herramientas que nos permiten establecer las formas como los datos circularán por el sistema, pasando por distintas entidades y siendo almacenadas en las bases de datos u archivos de salida. Es posible conceptualizar todo lo anterior a partir de diagramas de flujo de información y datos, además de la creación del diccionario de datos, donde se definirán los tipos de datos que el sistema podrá manejar y su estructura lógica en una o varias tablas ya en una base de datos.

Las estructuras de procesamiento de la información serán diseñadas a partir de los árboles o tablas de decisión que ayudan a establecer la estructura en que el sistema tomará sus decisiones con base en una serie de reglas derivadas de la construcción de dichas tablas o árboles. El empleo de diagramas de ruta crítica, adicionalmente, llevará a estimar los tiempos de cada actividad a realizar en las fases subsecuentes y de la misma fase de diseño.

En cuanto a la metodología de diseño, los ingenieros de *software* cuentan con diversas opciones que podrán seleccionar según las características del sistema que se esté desarrollando; incluso es posible combinar herramientas de cada una para obtener mejores resultados.

GLOSARIO DE LA UNIDAD

Diagrama

Representación gráfica en la que se muestran las relaciones entre las diferentes partes de un conjunto o sistema, o los cambios de un determinado fenómeno.

Diseño

Concepción original de un objeto u obra destinados a la producción en serie.

Entidad

En sentido general, se emplea para denominar todo aquello cuya existencia es perceptible por algún sistema animado. Una entidad puede, en consecuencia, ser concreta, abstracta, particular o universal.

Metodología

Hace referencia al conjunto de procedimientos racionales utilizados para alcanzar una gama de objetivos que rigen en una investigación científica, una exposición doctrinal o tareas que requieran habilidades, conocimientos o cuidados específicos. También se entiende como el estudio o elección de un método pertinente para un determinado objetivo.

Relación

En una base de datos relacional, todos los datos se almacenan y se accede a ellos por medio de relaciones. Las relaciones que almacenan datos se llaman "relaciones base" y su implementación, "tabla". Otras relaciones no almacenan datos, pero son calculadas al aplicar operaciones relacionales. Estas relaciones son "derivadas" y su implementación se conoce como "vista" o "consulta". Las relaciones derivadas son convenientes, ya que expresan información de varias relaciones que actúan como si fueran una sola.

Ruta crítica
Algoritmo utilizado para el cálculo de tiempos y plazos en la planificación de proyectos.

ACTIVIDADES DE APRENDIZAJE

ACTIVIDAD 1

Discute con tus compañeros el tema diagramas de flujo de datos vs. diagramas de flujo de información, con base en las siguientes preguntas:

1. ¿Cuál es la utilidad de los diagrama de flujo de datos y de información en el diseño de sistemas?
2. ¿Cuáles son las diferencias más importantes entre ambos diagramas?

ACTIVIDAD 2

Investiga un poco más acerca de los diccionarios de datos y sus características. Luego, con base en la información reunida, haz una síntesis donde expongas los elementos principales de un diccionario de datos. Incluye tus conclusiones; no olvides enunciar tus referencias.

ACTIVIDAD 3

Construye un árbol de decisiones simple que te permita ramificar dos posibles soluciones del problema de las ocho reinas en un tablero de ajedrez. Consulta el problema y sus posibles soluciones en http://ende.cc/agujero/juegos/8reinas_R.html

ACTIVIDAD 4

Elabora un cuadro comparativo sobre las diversas metodologías de diseño mencionadas en esta unidad. Destaca sus características principales y las herramientas empleadas en cada una.

ACTIVIDAD 5

Investiga sobre dos metodologías de diseño de sistemas diferentes a las estudiadas en la unidad. Luego, discute con tus compañeros las características de esas metodologías y sus principales aplicaciones. Opina acerca de las metodologías investigadas por dos de tus compañeros.

CUESTIONARIO DE REFORZAMIENTO

Responde las siguientes preguntas.

1. ¿Qué es la fase de diseño de sistemas?
2. ¿Cuál es la diferencia de un diagrama de flujo de información y uno de datos?
3. ¿Qué es el diccionario de datos y cuáles son sus principales características?
4. Menciona dos aplicaciones de los árboles de decisión.
5. Explica brevemente qué es una tabla de decisión.
6. ¿Qué son PERT y CPM?
7. Menciona dos características del análisis estructurado.
8. ¿Qué es el análisis estructurado?
9. ¿Cuáles son los diagramas empleados por la metodología HIPO?
10. ¿Qué es la metodología *bussines system planning*?

LO QUE APRENDÍ

Elabora un mapa conceptual sobre los elementos de la fase de diseño de sistemas. Incluye las metodologías y herramientas que pueden ser empleadas.

EXAMEN DE AUTOEVALUACIÓN

Responde verdadero (V) o falso (F).

1. El diseño de sistemas basa gran parte de sus principios en el diseño de *software*, que a su vez implica el uso de técnicas de diseño, codificación y prueba. ( )
2. El diagrama de flujo de datos (DFD) es la representación gráfica de la interacción de los usuarios con los diversos módulos del sistema. ( )
3. El diccionario de datos es un listado organizado de todos los elementos de datos pertinentes para el sistema, con definiciones precisas y rigurosas. ( )
4. La tabla de decisión es una matriz de renglones y columnas que indican condiciones y acciones a seguir en cierto proceso. ( )
5. Los árboles de decisión son una técnica que permite analizar decisiones aleatorias basadas en el uso de resultados y probabilidades asociadas. ( )
6. CPM supone que el tiempo para realizar cada una de las actividades es una variable aleatoria descrita por una distribución de probabilidad. ( )
7. PERT es un método basado en redes que tienen como principal objeto asistir en la planeación, programación y control de proyectos. ( )
8. El modelo de análisis estructurado es una técnica de ingeniería de *software* que utiliza diagramas y gráficos para representar y desarrollar las especificaciones del sistema, de modo que sean fácilmente entendibles por los usuarios. ( )
9. HIPO fue desarrollado por IBM como esquemas de representación para un desarrollo jerárquico de arriba abajo, y como una ayuda de documentación para productos comercializados. ( )
10. La metodología ISAC ayuda al diseño de estructuras de programas identificando la salida y resultado del procedimiento: trabaja hacia atrás para determinar los pasos y combinaciones de entrada necesarios para producirlos. ( )

MESOGRAFÍA

BIBLIOGRAFÍA RECOMENDADA

Autor	Capítulo	Páginas
Roger S. Pressman. *Ingeniería del* software*. Un enfoque práctico*	13	217-235
Vasconcelos, Jorge. *Manual de construcción de programas*.	2	12-29

BIBLIOGRAFÍA BÁSICA (REFERENCIAS)

Pressman, Roger. *Ingeniería del* software. *Un enfoque práctico,* 5.ª ed. España: McGraw-Hill, 2002, 601 pp.

Vasconcelos, Jorge. *Manual de construcción de programas*. México: Grupo Editorial Patria, 2000, 165 pp.

BIBLIOGRAFÍA COMPLEMENTARIA

1. Bardou, Louis. *Mantenimiento y soporte logístico de los sistemas informáticos.* México: Alfa Omega-Marcombo, 2004, 292 pp.
2. Bochinno, William A. *Sistemas de información para la administración, técnicas e instrumentos.* México: Trillas, 2002.
3. Bonsón, Enrique. *Tecnologías inteligentes para la gestión empresarial*. México: Alfa Omega-Rama, 2002, 258 pp.
4. Cornella, Alfons. *Información digital para la empresa, una introducción a los servicios de información electrónica.* México: Alfa Omega-Marcombo, 2004, 196 pp.
5. Lardent, Alberto R. *Sistemas de información para la gestión empresarial, procedimientos, seguridad y auditoría*. Buenos Aires: Pearson Education-Prentice Hall, 2001.
6. Levine, Guillermo. *Computación y programación moderna*. México: Addison Wesley, 2000.
7. Long, Nancy y Larry Long. *Introducción a las computadoras y a los sistemas de información, edición Internet.* México: Prentice Hall, 1999, 416 pp.
8. McLeod, Raymond, Jr. *Sistemas de información gerencial,* 7.ª ed. México: Prentice Hall, 1999, 688 pp.
9. Oz, Effy. *Administración de sistemas de información,* 2.ª ed. México: Thomson-Learning, 2001, 578 pp.
10. Peña R., Baeza-Yates, R. y Rodríguez, J. *Gestión digital de la información. de bits a bibliotecas digitales y la web.* México: Alfa Omega-Rama, 2004, 464 pp.
11. Piattini, Mario; J. Antonio Calvo-Manzano, Joaquín Cervera y Luis Fernández. *Análisis y diseño detallado de aplicaciones informáticas de gestión.* México: Alfa Omega-Rama, 2004, 728 pp.
12. Stair, Ralph M. *Principios de sistemas de información*, 4.ª ed. México: Thomson Learning, 2002, 692 pp.

13. Walker, D.W. *Sistemas e información para la administración.* México: Alfa Omega-Marcombo, 2001, 360 pp.

SITIOS ELECTRÓNICOS

Dirección electrónica	Descripción
http://goo.gl/gIhqlZ	Sistemas de inducción de árboles de decisión. Por Enrique Bonsón Ponte.
http://goo.gl/FNsJb6	Aprendizaje de árboles de decisiones. Por José M. Sempere, de la Universidad Politécnica de Valencia.
http://goo.gl/XG5MFH	Curso en línea de análisis y diseño de sistemas.
http://goo.gl/Isf5Vg	ISAC, por Fokkelien von Meyenfeldt.

UNIDAD 4
DESARROLLO DE SISTEMAS

OBJETIVO ESPECÍFICO

Al finalizar la unidad, el alumno reconocerá los elementos que conforman la fase de desarrollo de sistemas y su importancia en los diversos modelos de ciclos de vida de sistemas de información.

INTRODUCCIÓN

La fase de desarrollo de sistemas es la fase de construcción misma del sistema. En este momento, se realizan diversas actividades que dan vida al sistema de información en sí, comenzando por la asignación de roles tanto para la parte administrativa como para la técnica, donde cada persona involucrada en el desarrollo del sistema tomará un papel que, de manera individual o grupal, permitirá la construcción del sistema en su totalidad. Para el desarrollo correcto de un sistema de información es necesario identificar y seguir un modelo de ciclo de vida, y así determinar los productos finales que serán generados al final de cada fase y la forma de llevar a cabo cada tarea a la que se asocian.

Además, el seguimiento de una metodología de desarrollo es esencial para conseguir un sistema de calidad. Es decir, el sistema debe cumplir a cabalidad cada uno de los requerimientos planteados en la fase de análisis. En este orden, la unidad expone algunos modelos que pueden ser implementados por los desarrolladores de *software*.

Por último, revisaremos la fase de configuración, que permitirá al sistema interactuar de forma correcta con su entorno.

LO QUE SÉ

Escribe tu concepto personal de la fase de desarrollo de un sistema, agrega los elementos que consideres que la integran y describe brevemente su función.

TEMARIO DETALLADO (8 HORAS)

4.1. Principios de la ingeniería de *software*

4.2. Roles que intervienen en el desarrollo de sistemas

- 4.2.1. Roles técnicos
- 4.2.2. Roles administrativos

4.3. Tipos de ciclos de vida para el desarrollo de sistemas

- 4.3.1. Modelos conservadores
  - 4.3.1.1. Cascada
  - 4.3.1.2. Prototipo evolutivo
- 4.3.2. Modelos actuales
  - 4.3.2.1. *Personal software process*
  - 4.3.2.2. Proceso unificado
  - 4.3.2.3. *Team software process*

4.4. Fases que integran los ciclos de vida de los sistemas

- 4.4.1. Descripción de las fases
- 4.4.2. Roles que participan en cada fase
- 4.4.3. Productos que se generan en cada fase

4.5. Aplicación de la calidad en el desarrollo de sistemas

- 4.5.1. Importancia
- 4.5.2. Rol
- 4.5.3. Actividades
- 4.5.4. Tipos de modelos: CMM, ISO, SPICE

4.6. Aplicación de la configuración del *software* en el desarrollo de sistemas

- 4.6.1. Importancia
- 4.6.2. Rol
- 4.6.3. Actividades

4.1. Principios de la ingeniería de *software*

En palabras de Fritz Bauer, la ingeniería del *software* “es el establecimiento y uso de principios robustos de la ingeniería a fin de obtener económicamente *software* que sea fiable y que funcione eficientemente sobre máquinas reales”[32]. Esta definición es aceptada ampliamente en el campo de la informática, pero no nos dice directamente cuáles son esos principios robustos que menciona, o la forma de construir el *software* de forma económica.

Según Roger Pressman[33], la ingeniería de *software* se puede considerar como una tecnología multicapa como lo muestra la siguiente figura. Ello indica que, como cualquier enfoque de ingeniería, la ingeniería de *software* debe apoyarse en la organización para asegurar un enfoque de calidad en el desarrollo del *software*.

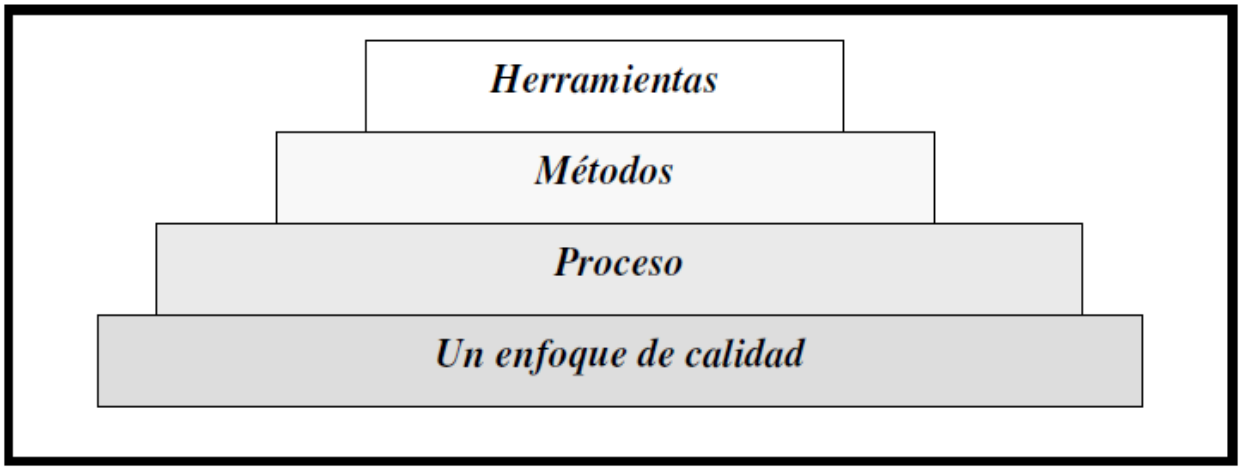


Figura 4.1. Capas de la ingeniería de *software*.[34]

La ingeniería de *software* tiene su fundamento en la capa de proceso, pues dentro de esta capa se unen las tecnologías y técnicas necesarias que permiten tener un

[32] Pressman, R. *Ingeniería del* software, 4.ª ed. McGraw-Hill, pp. 17-19.
[33] Pressman, R. *Ingeniería del* software, 4.ª ed. McGraw-Hill, pp. 17-19.
[34] En Pressman, R. *Ingeniería del* software, 4.ª ed. McGraw-Hill, pp. 17-19.

desarrollo racional y oportuno de la ingeniería de *software*. En la capa de proceso, se define el marco de trabajo para todas las áreas involucradas en el desarrollo de un proyecto de desarrollo de *software*, marco que permite la entrega efectiva de la tecnología asociada a la ingeniería de *software.*

La capa de métodos indica la forma técnica de construir *software*. Los métodos de la ingeniería de *software* son variados y abarcan una gran cantidad de tareas: análisis de requisitos, diseño y construcción de *software*, pruebas y mantenimiento.

La capa de herramientas proporciona el soporte para la capa de proceso y métodos. Pueden ser herramientas que nos proporcionen ese soporte de forma automática o semiautomática; es decir, la información generada por una herramienta puede ser utilizada por otra, y así sucesivamente, lo que genera un soporte para la ingeniería de *software*, la ingeniería de *software* asistida por computadora *(computer-aided softwareengineering case).*

Las herramientas CASE combinan *software*, *hardware* y una base de datos con información sobre el análisis, diseño, construcción y pruebas de *software* que ayudan al entorno de la ingeniería de *software* denominado CAD/CAE o diseño /ingeniería asistida por computadora (*computer-aide desing/engineering).*

Para llevar a cabo la ingeniería de *software*, es necesario establecer un método de desarrollo dividido en tres fases[35]:

- *Definición.* Responde a varias interrogantes: ¿qué información ha de ser procesada?, ¿qué función y rendimiento se desea?, ¿qué comportamiento del sistema?, ¿qué interfaces van a ser establecidas?, ¿qué restricciones

[35] Apuntes digitales de la asignatura de Desarrollo de sistemas. UPIICSA, Instituto Politécnico Nacional. Disponible en http://goo.gl/wqFX2h. Recuperado: 08/07/2013.

de diseño existen?, ¿qué criterios de validación se necesitan para definir un sistema correcto?

- *Desarrollo*. Responde a las siguientes interrogantes: ¿cómo han de diseñarse las estructuras de datos?, ¿cómo ha de construirse la función como una arquitectura del *software*?, ¿cómo han de implantarse detalles sobre los procedimientos?, ¿cómo han de caracterizarse las interfaces?, ¿cómo ha de traducirse el diseño en el lenguaje de programación?, ¿cómo deben realizarse las pruebas? Sea cual fuere el método empleado para determinar las cuestiones anteriores, el resultado siempre será el diseño del *software*, la generación de código y la prueba del *software*.

- *Mantenimiento.* Esta fase se enfoca en la detección de errores, mejora y adaptación del *software* de acuerdo con su entorno y los nuevos requisitos definidos por los clientes. Retoma algunas tareas de la fase de definición, ya que existirán casos donde se deban realizar modificaciones mayores al diseño inicial del *software.*

4.2. Roles que intervienen en el desarrollo de sistemas

Cuando se desarrollan sistemas de información, es necesaria la cooperación de varios individuos que ayudarán a su creación, desde los clientes, quienes establecerán los objetivos que debe de cumplir el sistema, hasta los involucrados en su formación, como analistas, programadores, capturitas, *testers*, etcétera. Cada uno tendrá a su cargo una o varias tareas para el desarrollo de calidad del sistema de información. Así, el desarrollo de sistemas de información es una actividad grupal, donde se conjugan las distintas capacidades de los integrantes del grupo para alcanzar los objetivos establecidos en los términos pactados. Por ello, a cada miembro se le asignará un rol, según sus capacidades y experiencia en el desarrollo de sistemas. Dependiendo en

gran medida de la magnitud del proyecto, los roles podrán extenderse a más personas o podrán ser integradas en una sola persona de ser necesario. En todo caso, es importante que todos los participantes de un proyecto de desarrollo de *software* tengan perfectamente claros sus roles y responsabilidades, lo que evitará conflictos y redundará en un trabajo eficaz.

4.2.1. Roles técnicos

Debemos entender que el desempeño de roles es fundamental para el logro exitoso de un proyecto de desarrollo de *software*, en particular los relacionados con el área técnica:

- *Programadores.* Son responsables de convertir una especificación en una operación funcional a través del uso de uno o varios lenguajes de programación.

- *Diseñadores*. Encargados de generar el diseño del sistema.

- *Evaluadores (testers)*. Personal encargado de detectar fallas en las diversas etapas del desarrollo del *software*.

- *Aseguradores de calidad.* Responsable de lograr que el *software* desarrollado cumpla los estándares establecidos en normas internacionales de calidad.

- *Documentadores.* Tienen a su cargo generar la documentación que acompañará al sistema. Recaban la información generada desde la fase de análisis hasta los resultados de la puesta en marcha y la bitácora de mantenimiento. Los documentos producidos en cada fase son facilitados al resto de los miembros del equipo de desarrollo para mantener coherencia con el trabajo realizado.

- *Ingeniero de validación y verificación.* A diferencia de los aseguradores de calidad, el ingeniero de validación es responsable por la evaluación del *software* al final de su proceso de desarrollo para asegurarse que está libre de fallas y cumple con sus requisitos.

- *Administrador de configuración.* Se encarga de realizar las adecuaciones necesarias a los equipos de cómputo donde será instalado el sistema para su funcionamiento correcto. "La administración de la configuración de *software* corresponde a la administración de la configuración aplicada a un sistema, o a partes de un sistema, predominantemente correspondiente a *software*".[36]

4.2.2. Roles administrativos

Los roles administrativos son los encargados de administrar y controlar los recursos asignados al proyecto en cada una de sus fases; y tienen como propósito que se cumplan los objetivos en los tiempos asignados y de forma correcta. En cuanto a los recursos destinados a un proyecto de desarrollo de sistemas de información, pueden ser financieros, humanos, tecnológicos, etcétera, según la naturaleza del proyecto.

Sin excepción, todo proyecto profesional de desarrollo de sistemas contará con un administrador de proyecto, quien será responsable de la administración general de los recursos asignados al proyecto a partir de los siguientes objetivos[37]:

- Tener el producto a tiempo, bajo presupuesto y con los requisitos de calidad definidos.

[36] Fuller-Padilla, D. Apuntes de Taller de ingeniería de *software.* Disponible en http://goo.gl/rOmMzT. Recuperado: 08/07/2013.
[37] Camargo, L. R. y Rojas, A. R. *Gestión de proyectos orientados a la web*. Universidad de los Llanos. Disponible en http://goo.gl/iKx9kP. Recuperado: 08/07/2013.

- Terminar el proyecto con los recursos asignados.
- Coordinar los esfuerzos generales del proyecto, ayudando a cada uno de sus integrantes a cumplir sus objetivos particulares. Al final, se cumplirá el objetivo general.
- Cumplir con éxito las diferentes fases de un proyecto utilizando herramientas de administración.
- Cumplir con las expectativas del cliente.

Algunos objetivos específicos a cumplir por un administrador de proyecto:

- Comenzar y terminar cada actividad según lo planificado.
- Lograr el mejor uso de los recursos disponibles.
- Observar cada actividad para detectar y resolver inconvenientes.

La comunicación es muy importante dentro de cualquier proyecto. Por eso, el administrador de proyecto tendrá una comunicación fluida y eficiente con el resto de los miembros del equipo de desarrollo a fin de mantenerse informado de los avances de cada integrante, de los problemas generales y particulares que se van presentando, y para la toma de decisiones dentro del proyecto.

4.3. Tipos de ciclos de vida para el desarrollo de sistemas

La creación de un sistema de información es un proceso cronológico que parte desde su concepción hasta su puesta en marcha, y mejora a través del mantenimiento. Tiene diversas etapas que dan vida al sistema mismo y, a través de ellas, es posible verificar y validar el cumplimiento de cada uno de los requisitos establecidos que garantizan la calidad del sistema. La conjunción de estas etapas de desarrollo se conoce como “ciclo de vida de un sistema de información”, el cual, dependiendo de la metodología

empleada, será más corto o prolongado; pero siempre iniciará con una etapa de análisis, pasando por una de diseño, desarrollo, implementación y mantenimiento.

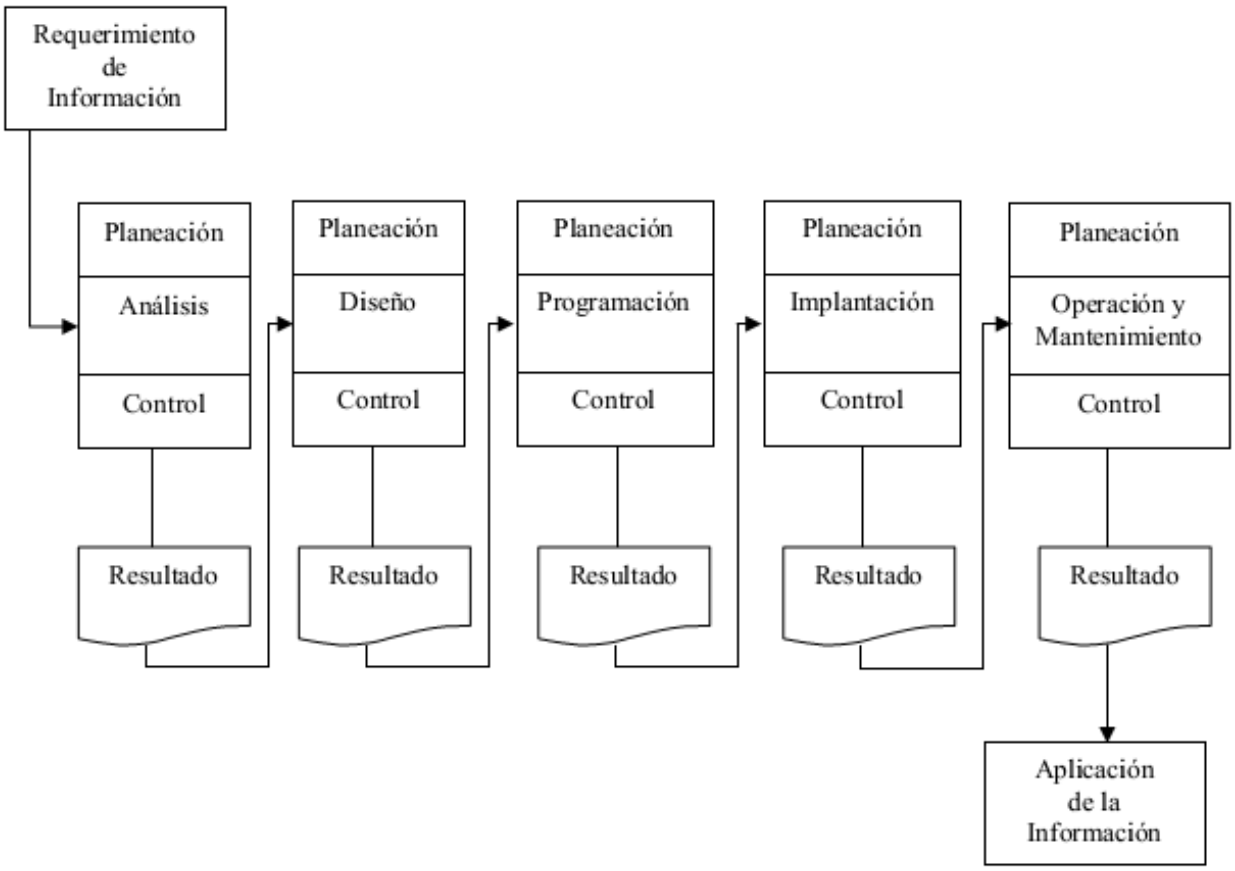


Figura 4.2. Proceso de desarrollo de sistemas de información[38].

[38] Peña Ayala, A. *Ingeniería de* software. *Una guía para crear sistemas de información.* Instituto Politécnico Nacional. México, 2006, p. 30.

4.3.1. Modelos conservadores

Al principio, los desarrolladores de *software* no tenían una metodología estandarizada, trabajaban por puro instinto; es decir, era un desarrollo convencional o artesanal. Lo que caracterizaba a este tipo de *software* era su estructura monolítica: consistía en un solo programa fuente principal que realizaba todas las funciones solicitadas; se carecía de flexibilidad a los cambios, lo que obligaba a un remplazo total del *software* cuando se requería alguna modificación.

Al pasar de los años y con la creciente necesidad de llevar un control sobre el desarrollo del *software*, surge el paradigma de la programación estructurada, lo que se convertiría a la postre en los comienzos de la ingeniería de *software*. Como en la mayor parte de los desarrollos informáticos, la programación estructurada aparece en la década de 1960, con fines científicos; y pasando al ámbito público en la década de 1970, con la expansión de las computadoras hacia las empresas.

La programación estructurada incorpora el establecimiento y uso de normas para el desarrollo de sistemas; asimismo, otorga al ciclo de vida de los sistemas de información la fase de diseño y crea técnicas estructuradas tanto para los aspectos técnicos como para los de construcción de *software*.

En 1975, Myers, Yourdon y Constantine[39] definen la estructura modular en la programación estructurada. Consideran a cada módulo de un programa como el componente básico en el desarrollo de *software*, normalizando los procesos y refinando el funcionamiento del *software*. Estos son los primeros pasos para el alcanzar el desarrollo de *software* de calidad.

[39] Pressman, R. *Ingeniería del* software, 4.ª ed. McGraw-Hill, p. 246.

El paradigma de la programación estructurada fundamenta el desarrollo de *software* en un diseño de tipo *top-down* o descendente, poniendo atención especial en las especificaciones funcionales. La programación estructurada emplea diagramas con etiquetas o texto de referencia, con el fin de que los diagramas resultantes puedan ser interpretados de manera independiente, minimizando la redundancia de la información, lo que permite un diseño más flexible a los cambios con impactos mínimos en el diseño del sistema.

Durante la década de 1980, nace el paradigma de programación orientada a objetos, que da un trato diferente a los procesos y datos de los sistemas, agrupándolos de tal forma que se pueda dar un tratamiento modular tanto a la información como al procesamiento de la misma. Uno de los primeros lenguajes de programación orientados a objetos fue Smalltalk, que pone énfasis en la abstracción de los datos, tratando a los problemas como un conjunto de objetos de datos que tenían asociados una serie de operaciones a ellos. Smalltalk fundamenta su operación en la abstracción, modularidad y flexibilidad del manejo de la información derivados del paradigma estructurado.

Podemos decir que las metodologías empleadas en la programación orientada a objetos nacen de los principios de diseño de programación estructurada.

4.3.1.1. Cascada

El modelo en cascada es un enfoque de diseño de sistemas donde el sistema debe pasar por una serie de pasos secuenciales y lineales que van desde el análisis de la información hasta su puesta en marcha y mantenimiento. Se conforma de las siguientes etapas[40]:

- Análisis de requisitos del sistema
- Análisis de requisitos del *software*
- Diseño preliminar
- Diseño detallado
- Codificación y pruebas
- Explotación (u operación) y mantenimiento

Características del modelo:

- Cada fase empieza cuando se ha terminado la anterior.
- Para pasar a la fase posterior es necesario haber logrado los objetivos de la previa.
- Es útil como control de fechas de entregas.
- Al final de cada fase, el personal técnico y los usuarios tienen oportunidad de revisar el progreso del proyecto.

[40] Sommerville, I. *Ingeniería del* software, 7.ª ed., p. 62.

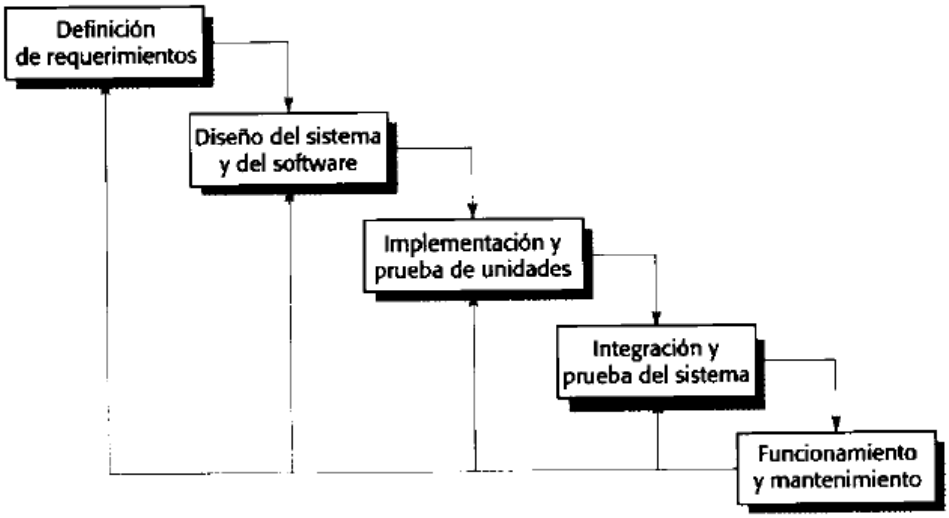


Figura 4.3. Modelo de desarrollo en cascada.[41]

4.3.1.2. Prototipos evolutivos

El método de prototipos evolutivos, o evolutivo, se basa en desarrollar un modelo inicial del sistema, poniéndolo a consideración de los clientes y enriqueciendo su funcionamiento a través de los comentarios de los mismos, las revisiones de las especificaciones y el desarrollo de nuevas versiones que incorporen los puntos de vista obtenidos. El proceso se repite hasta llegar a un prototipo adecuado, el cual se convertirá en la versión final del sistema.

En el modelo de prototipos, el ciclo de vida de los sistemas se modifica dando como resultado las siguientes fases:

- Análisis de requisitos del sistema
- Análisis de requisitos del *software*
- Diseño, desarrollo e implementación del prototipo-prueba del prototipo

41. Sommerville, I. *Ingeniería del* software, 7.ª ed., p. 62.

- Refinamiento iterativo del prototipo
- Refinamiento de las especificaciones del prototipo
- Diseño e implementación del sistema final
- Explotación (u operación) y mantenimiento

Este proceso se esquematiza en la siguiente figura:

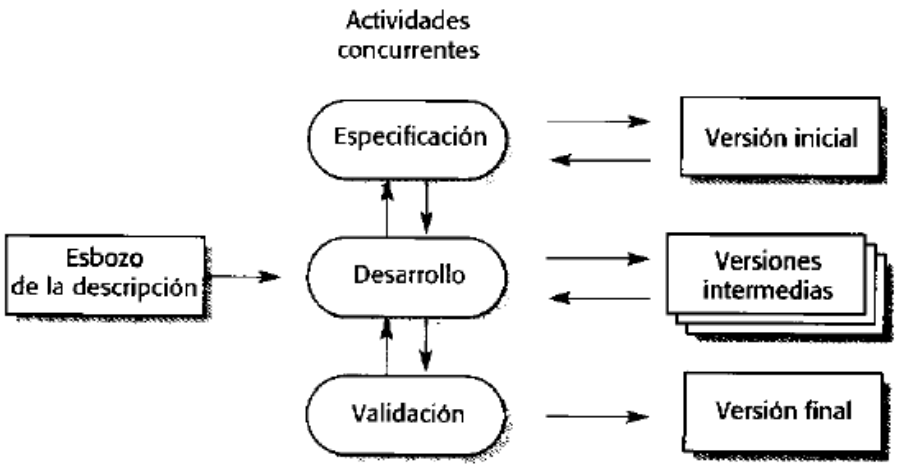


Figura 4.4. Modelo evolutivo.[42]

[42]. Sommerville, I. *Ingeniería del* software, 7.ª ed., p. 63.

4.3.2. Modelos actuales

En los métodos de programación, algunos elementos del *software* son reutilizables, por ello, en ocasiones se tienen partes de código similares que son incorporadas en diferentes módulos de un sistema de información. El concepto de reutilización de código nace del método evolutivo y es parte fundamental para el desarrollo de las metodologías actuales enfocadas al desarrollo rápido de sistemas.

El paradigma de programación orientado a objetos permite la reutilización de los objetos que conforman un sistema de tal modo que el desarrollo de sistemas se desarrolla de forma iterativa e incremental. Algunos modelos que emplean esta filosofía se describen a continuación.

Modelo de agrupamiento o de clúster

Este modelo agrega a la fase de validación una fase de generalización de la información que permite agrupar las clases asociadas a un objeto en común, con lo que se crean pequeños ciclos iterativos que posibilitan la evolución del proyecto, donde cada nueva fase depende de lo desarrollado en la fase anterior. El modelo de agrupamiento tiene un enfoque de desarrollo ascendente que parte de las clases básicas del sistema.

Modelo fuente

Desarrollado por Henderson-Sellers y Edwards (1990)[43], representa un modelo con un alto grado de iteración entre cada fase tomando como base el análisis de los requisitos

[43] Modelos orientados a objetos. Curso FPE de analista funcional. Disponible en http://goo.gl/nX9ah2. Recuperado: 08/07/2013.

funcionales. A partir de los requisitos, se va desarrollando cada fase del ciclo de vida, comenzando una nueva iteración al retomar las clases que se encuentran alojadas en un repositorio o piscina.

Modelo *pinball*

Este modelo toma como referencia al juego de *pinball* como la forma de desarrollar un sistema; el sistema es representado por la pelota y los desarrolladores del sistema son los jugadores. El modelo comienza por buscar clases, atributos, métodos y relaciones entre objetos de forma iterativa; así, el modelo permite a los desarrolladores definir las colaboraciones, herencia entre objetos, agregación y módulos o subsistemas que integran el proyecto. Determinado lo anterior, se procede a la programación, a realizar las pruebas y la implementación del sistema. Así como en el juego, las fases del ciclo de vida pueden realizarse en cualquier orden o de forma simultánea, por lo que la habilidad y experiencia de los jugadores (desarrolladores) son cruciales.

A partir de la década de 1990, con el crecimiento en la demanda de desarrollo de sistemas, el Instituto de Ingeniería Eléctrica y Electrónica (IEEE) estableció el estándar IEEE 1074-1991, donde se detallan las fases del ciclo de vida base en el desarrollo de *software* y la documentación que debe generarse en cada fase.

4.3.2.1. *Personal software process*

El Instituto de Ingeniería de Software (SEI) desarrolló hace poco una metodología para el desarrollo de *software*, el *personal software process* (PSP), que busca mejorar la

forma como se crea *software* con miras a llegar a una calidad del mismo, a partir de su planeación, costos y productividad.

El PSP emplea un conjunto de documentos o *scripts*, donde se detallan las actividades y tareas que los desarrolladores deben realizar durante todo el proceso de creación del proyecto de *software*; estos *scripts* son el punto central del PSP. Una de las ventajas que proporciona a los desarrolladores el empleo de *scripts* es la generación de datos principalmente estadísticos, que sirven para identificar las fortalezas y debilidades del proyecto y poder mejorarlo.

Adicionalmente al empleo de los *scripts*, PSP basa su metodología en la estimación, la cual permite establecer periodos en los cuales se deben desarrollar las tareas y actividades de un cierto proceso. Esto da pie a un mejor control sobre el tiempo de desarrollo total del proyecto y sus costos. La información generada por las estimaciones se aprovecha en especial para evaluar los procesos actuales y realizar avances en los procesos futuros, lo que ayuda a los desarrolladores a identificar sus propias fortalezas y debilidades, lo que además implica una oportunidad para pulir sus habilidades y mejorar la conformación del *software*.

La metodología PSP implica un análisis de requisitos previo, pues éstos no están incorporados en las tareas y actividades definidas en los *scripts*.

4.3.2.2. Proceso unificado (RUP, *rational unified process*)

RUP es un proceso de desarrollo de *software* que, junto con el lenguaje unificado de modelado (UML), constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos[44]. Su objetivo es

[44] Metodologías de desarrollo de *software*. Apunte desarrollado por Grupo CUYS de Facultad de Ciencias Exactas, UNICEN Argentina. Disponible en http://goo.gl/Pkfvrl. Recuperado: 08/07/2013.

asegurar la producción de *software* de calidad en plazos y presupuestos predecibles. Basa su metodología en el empleo de casos de uso centrados en la arquitectura del proyecto. Trabaja de forma iterativa generando prototipos, y de forma incremental al aprobar un prototipo convirtiéndolo en una versión del proyecto.

El ciclo de vida que integra al RUP consta de cuatro fases: planificación o concepción, elaboración, construcción y transición. Y como ya se mencionó, trabaja de manera iterativa y al final de cada iteración se genera un prototipo funcional que se evalúa y es sometido a una mejora; luego, se comienza de nuevo con el ciclo de vida. La siguiente figura ilustra la forma de trabajo de RUP en forma general.

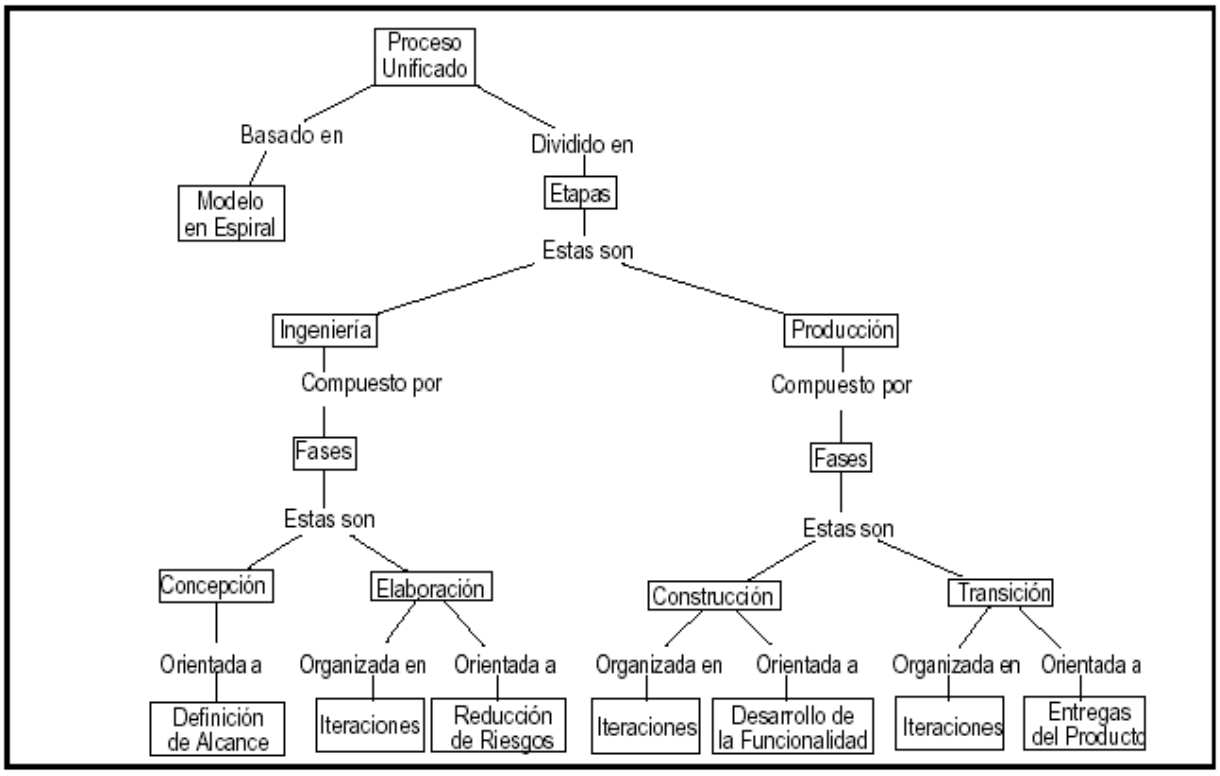


Figura 4.4. Estructura de RUP.

4.3.2.3. *Team software process*

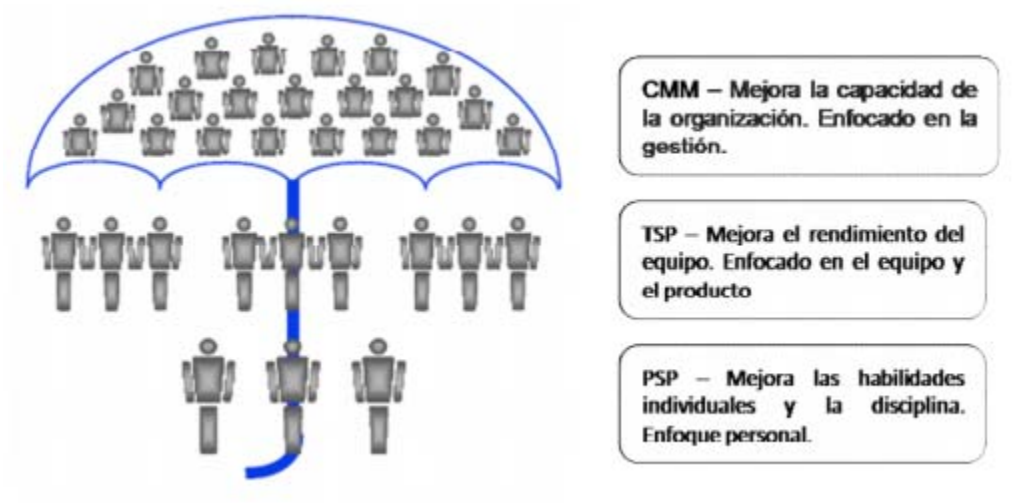


Figura 4.5. Métodos de mejora de proceso[45].

El *team software process* (TSP) es una metodología orientada al trabajo para el desarrollo de *software* y fomenta el trabajo en equipo de manera efectiva y natural. Su objetivo principal es la mejora de la calidad y productividad de un proyecto de desarrollo de *software*, con la intención de establecer acuerdos que optimicen los costos y tiempos de desarrollo del *software*. Y al ser una metodología orientada al desarrollo de *software* en equipos de trabajo, requiere, en primera instancia, la capacitación en *personal software process* (PSP) de los elementos que conformarán dichos equipos; así, los integrantes se acostumbrarán a trabajar con planes detallados.

TSP define los pasos necesarios para que se establezca un clima eficiente de trabajo en equipo. Para esto se apoya en el PSP, a fin de que cada miembro de los equipos sea responsable de la calidad de su trabajo, lo que permitirá al TSP establecer y mantener equipos autodirigidos. En consecuencia, TSP y PSP son metodologías complementarias: a través de PSP, TSP pueden establecerse actividades y tareas que

[45] Roy Steeven Yarce, D. *Uso de tecnologías y metodologías de desarrollo manejados en Pragma S.A. para la construcción de portales web.* Disponible en http://goo.gl/LNlDJV. Recuperado: 28/08/2013.

permitan al equipo de desarrollo mejorar los procesos de creación de *software* y garantizar su calidad[46].

- Con PSP, los desarrolladores utilizan procesos definidos y medibles. Se toma información de tamaño, tiempo y defectos al momento de realizar el trabajo, y los datos reunidos sirven para planear y monitorear el trabajo, administrar la calidad de los productos y medir y mejorar el desempeño.

- TSP ha permitido resolver problemas típicos de un negocio, como predicciones de costo y tiempo, mejora de productividad y ciclos de desarrollo, y mejora de calidad de productos.

- PSP/TSP aumenta el desempeño tanto de equipos como individuos; es disciplinado y ágil; provee beneficios inmediatos y medibles; acelera las iniciativas de mejora de procesos organizacionales.

- Con TSP, los equipos encuentran y reparan defectos en etapas tempranas del proceso de desarrollo. Lo que reduce de manera importante el tiempo de pruebas, y con una etapa de pruebas más corto, el ciclo completo se aminora.

[46] Metodología TSP/PSP, por Grupo de tecnologías Kernel. Disponible en http://goo.gl/5TrE24. Recuperado: 08/07/2013.

4.4. Fases que integran los ciclos de vida de los sistemas

El ciclo de vida de un sistema de información o de *software* describe las fases por las que pasa, desde su concepción hasta su formación. Cada fase comprende el desarrollo de diversas actividades que deben realizarse para llegar a un sistema funcional que cumpla con las expectativas de los clientes.

Las fases que tradicionalmente integran al ciclo de vida son las siguientes:

1. Requerimientos
2. Análisis/diseño
3. Construcción
4. Pruebas
5. Puesta en marcha/mantenimiento

4.4.1. Descripción de las fases

Requerimientos

Es la fase fundamental del ciclo de vida, donde se determinan los objetivos del sistema acordes a los de la organización, y se establecen la planificación del sistema y las estrategias a seguir para ajustar los procesos del negocio y los flujos de información de la organización. De esta fase surge la información necesaria para la creación del sistema de información que determinarán su funcionamiento y cómo se integrará a la organización.

Análisis/diseño

Durante esta fase, se concreta la arquitectura funcional del sistema, partiendo de los requerimientos obtenidos de la fase anterior, y se efectúan una serie de actividades que permiten construir de forma conceptual el sistema de información. En esta fase se generan documentos como el diccionario de datos, diagramas de flujo de datos, diagramas entidad-relación, etcétera.

Construcción

Esta fase toma la estructura conceptual generada en la etapa anterior y la convierte en un sistema funcional. Aquí, se construye el *software* que dará vida al sistema de información y se define la estructura lógica que lo acompañará en su funcionamiento (base de datos, lenguajes de programación, sistemas operativos, etcétera).

Pruebas

Como todo lo construido por los seres humanos, el *software* es perfectible. En este orden, la fase de pruebas está orientada a someter al sistema a condiciones de trabajo que lo lleven al límite de sus capacidades, con el propósito de detectar problemas en su funcionamiento y corregirlos antes de ponerlo a disposición de los usuarios.

Puesta en marcha/mantenimiento

Esta fase comienza con la liberación del sistema, es decir, lo pone a disposición de los usuarios finales para que lo integren a sus actividades diarias de trabajo. Realizado lo anterior, será necesario monitorear su funcionamiento para detectar errores que no

fueron percibidos en la fase de pruebas y realizar las correcciones necesarias a través de actualizaciones del sistema. En este momento también se ejecutan actividades enfocadas a mantener la funcionalidad del sistema y la disponibilidad y seguridad de la información que este maneja.

4.4.2. Roles que participan en cada fase

Administrador de proyecto

Es el encargado de administrar los recursos asignados al proyecto y verificar que todas las actividades se efectúen adecuadamente en tiempo y forma, por lo que se encuentra presente a lo largo de todas las fases del ciclo de vida del sistema.

Analistas

Se encargan de traducir lo que el cliente quiere que realice el sistema, en los requerimientos funcionales y no funcionales que ayudarán a la conceptualización del sistema de información y a su posterior construcción. Lo anterior indica que deberán trabajar de forma cercana con el cliente, en especial durante la fase de requerimientos del ciclo de vida. Asimismo, estarán presentes en la fase de diseño, pues a través de ellos los desarrolladores podrán modelar el sistema de información deseado y llevarlo a su construcción.

Diseñadores

Son responsables de establecer el diseño arquitectónico del sistema, así como el diseño a detalle de sus componentes. Por lo general, están presentes a partir de la fase de diseño hasta la puesta en marcha y mantenimiento del sistema.

Programadores

Se encargan de convertir los diseños arquitectónicos y detallados del sistema en código fuente de programación, con el lenguaje de programación seleccionado para ello. Adicionalmente, se pueden apoyar en otras herramientas de *software* (CASE) para generar el código fuente.

Ténsters

Diseñan y aplican las pruebas a los prototipos o al sistema ya en su totalidad, con el objetivo de detectar fallas y validar que cumplan con los requerimientos establecidos. Esta función puede ser realizada por los mismos diseñadores o programadores.

Ingeniero de validación y verificación

Se encarga de verificar que sean cubiertos los requerimientos funcionales y no funcionales establecidos en la fase de requerimientos, para garantizar que las expectativas de los clientes sean cumplidas y asegurar la calidad del sistema desarrollado.

Documentadores

La documentación es importante en el ciclo de vida de un sistema de información, pues a través de ella será posible analizar el sistema para realizarle mejoras y entender su funcionamiento, así como para garantizar que el sistema no dependa de un solo equipo de desarrollo. En este orden, los documentadores se encargan de recabar toda la documentación generada durante el ciclo de vida del sistema y concentrarla en el manual técnico del sistema (los documentadores pueden ser miembros del equipo de desarrollo que se encuentren presentes a lo largo de todo el proceso de desarrollo del sistema).

Ingenieros de manutención

Llevan a cabo las tareas de mantenimiento preventivas y correctivas, con el objetivo de conservar la funcionalidad del sistema en forma óptima. Estos ingenieros pueden ser parte del equipo de desarrollo o personal capacitado por la organización para realizar dichas tareas, y por lo general se hacen presentes una vez puesto en marcha el sistema hasta el término de la vida útil de éste.

Clientes

Representan los intereses de una organización y se encargan, junto con los analistas y desarrolladores, de determinar las características del sistema de información que será construido y de que se apegue a los objetivos y estrategias establecidos por la organización. Independientemente del modelo de desarrollo seleccionado, el cliente está presente a lo largo de todo el ciclo de vida del sistema de información.

4.5. Aplicación de la calidad en el desarrollo de sistemas

La calidad del *software* se define como la "concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados, y con las características implícitas que se espera de todo *software* desarrollado profesionalmente"[47]. De esta definición, podemos inferir que un *software* de calidad siempre deberá cumplir a cabalidad con los requisitos funcionales y no funcionales establecidos por los clientes, y estará documentado con una metodología o una estandarización precisas, para garantizar que realice sus funciones de forma eficiente y óptima.

De acuerdo con Pressman, la calidad del *software* se basa en tres puntos fundamentales[48]:

1. Los requisitos del *software* son la base de las medidas de la calidad. La falta de concordancia con los requisitos es una falta de calidad.
2. Los estándares especificados definen un conjunto de criterios de desarrollo que guían la forma como se aplica la ingeniería del *software*. Si no se siguen esos criterios, casi siempre habrá falta de calidad.
3. Hay un conjunto de requisitos implícitos que a menudo no se mencionan (por ejemplo, el deseo por facilitar el uso y un buen mantenimiento). Si el *software* se ajusta a los requisitos explícitos, pero falla en alcanzar los requisitos implícitos, la calidad del *software* queda en entredicho.

[47] Pressman, R. *Ingeniería del* software, 5.ª ed., p. 131.
[48] Pressman, R. *Ingeniería del* software, 5.ª ed., p. 135.

1.5.1. Importancia

Basándonos en la definición de calidad del punto anterior, la garantía de calidad del software (SQA) es un “patrón de acciones planificado y sistemático”[49] requerido para asegurar la calidad del *software*. Esta garantía de calidad del *software* debe involucrar a todos los miembros de la organización que tengan alguna responsabilidad hacia el sistema, desde los directivos hasta los usuarios finales, quienes puntualizan, junto con los desarrolladores, los requisitos del sistema que son la base de la calidad del mismo.

1.5.2. Rol

El establecimiento de un grupo de SQA pretende ayudar a los desarrolladores de *software* en la construcción de un sistema de alta calidad que cumpla con los requisitos establecidos y los objetivos organizacionales.

4.5.3. Actividades

De acuerdo con las recomendaciones del Instituto de Ingeniería de Software (SIE), las actividades para establecer un plan de SQA deben englobar la planificación para la garantía de la calidad, supervisión, mantenimiento de registros, análisis y generación de informes.

[49] Pressman, R. *Ingeniería del* software, 5.ª ed., p. 135.

El SIE determina la manera de establecer un plan de SQA[50], como se describe a continuación.

- Establecimiento de un plan de SQA para un proyecto. El plan se desarrolla durante la planificación del proyecto y es revisado por todas las partes interesadas. Las actividades de garantía de calidad realizadas por el equipo de ingeniería del *software* y el grupo SQA son gobernadas por el plan. El plan identifica lo siguiente:

  - ❖Evaluaciones a realizar
  - ❖Auditorías y revisiones a realizar
  - ❖Estándares que se pueden aplicar al proyecto
  - ❖Procedimientos para información y seguimiento de errores
  - ❖Documentos producidos por el grupo SQA,
  - ❖Realimentación de información proporcionada al equipo de proyecto del *software*

- Participación en el desarrollo de la descripción del proceso de *software* del proyecto. El equipo de ingeniería del *software* selecciona un proceso para el trabajo que se va a realizar. El grupo de SQA revisa la descripción del proceso para ajustarse a la política de la empresa, los estándares internos del *software*, los estándares impuestos externamente (por ejemplo, 1SO 9001), y a otras partes del plan de proyecto del *software*.

- Revisión de las actividades de ingeniería del *software* para verificar su ajuste al proceso de *software* definido. El grupo de SQA identifica, documenta y sigue la pista de las desviaciones desde el proceso y verifica que se han hecho las correcciones.

[50] Pressman, R. *Ingeniería del* software, 5.ª ed., p. 136.

- Auditoría de los productos de *software* designados para verificar el ajuste con los definidos como parte del proceso del *software*. El grupo de SQA revisa los productos seleccionados; identifica, documenta y sigue la pista de las desviaciones; verifica que se han hecho las correcciones e informa periódicamente de los resultados de su trabajo al gestor del proyecto.

- Asegurar que las desviaciones del trabajo y los productos del *software* se documentan y manejan de acuerdo con un procedimiento establecido. Las desviaciones se pueden encontrar en el plan del proyecto, en la descripción del proceso, en los estándares aplicables o en los productos técnicos.

- Registrar lo que no se ajuste a los requisitos e informar a sus superiores. Los elementos que no se ajustan a los requisitos están bajo seguimiento hasta que se resuelven.

4.5.4 Tipos de modelos: CMMI, ISO, SPICE

(OJO, REVISAR EL ÍNDICE, CÓMO SE PLANTEA…)

ISO (creo que éste es el segundo)

La Organización Internacional para la Estandarización (ISO, por sus siglas en inglés) se encarga de promover el desarrollo de normas internacionales para la fabricación, comercio y comunicación para todas las ramas industriales, con excepción de la eléctrica y electrónica, y cuenta con representación en 164 países del mundo. La norma específica que atiende los aspectos de calidad es la ISO 9001, diseñada para la gestión

y aseguramiento de la calidad, especificación de los requisitos básicos para el desarrollo, producción, instalación y servicio a nivel de sistema y a nivel de producto.[51]

Estructura de ISO 9001:2008[52]

- Cap. 1 al 3. Guías y descripciones generales.
- Cap. 4. Sistema de gestión. Contiene los requisitos generales y requisitos para gestionar la documentación.
- Cap. 5. Responsabilidades de la dirección. Contiene los requisitos que debe cumplir la dirección de la organización: definir la política, asegurar que las responsabilidades y autoridades están definidas, aprobar objetivos, el compromiso de la dirección con la calidad, etcétera.
- Cap. 6. Gestión de los recursos. La norma distingue tres tipos de recursos sobre los cuales se debe actuar: RRHH, infraestructura y ambiente de trabajo. Aquí, se enuncian los requisitos exigidos en su gestión.
- Cap. 7. Realización del producto/servicio. Comprende los requisitos puramente de lo que se produce o brinda como servicio (la norma incluye servicio cuando denomina "producto"), desde la atención al cliente hasta la entrega del producto o el servicio.
- Cap. 8. Medición, análisis y mejora. En éste, se sitúan los requisitos para los procesos que recopilan y analizan la información, y que actúan en consecuencia. El objetivo es mejorar continuamente la capacidad de la organización para suministrar productos o servicios que cumplan con los requisitos. El objetivo declarado en la norma es que la organización busque sin descanso la satisfacción del cliente a través del cumplimiento de los requisitos.

[51] Concepto de ISO-9001. Sitio oficial de la ISO. Disponible en http://goo.gl/6dX8eM. Recuperado: 28/08/2013.
[52] Norma internacional ISO 9001**.** Sistema de gestión de calidad. Requisitos. Traducción disponible en http://goo.gl/zFCLo3. Recuperado: 08/07/2013.

Así, la estructura de la ISO 9001:2008 permite su aplicación en cualquier actividad productiva, sin importar el tipo de producto o servicio que se brinde.

CMMI

El modelo de madurez de *software* CMMI sirve como base en las organizaciones dedicadas el desarrollo de *software*, para aumentar el desempeño de sus procesos de desarrollo ayudándoles en la evaluación para alcanzar la madurez de los mismos, y en la implementación de estrategias de mejora continua.

CMMI tiene dos representaciones del modelo: continua (capacidad de cada área de proceso) y por etapas (madurez organizacional). Dentro de la primera, se establece una línea base que servirá para medir el grado de mejora de cada tarea del proceso; y al igual que en la representación por etapas, se manejan las prácticas que, dentro de este enfoque, ayudan al crecimiento y la mejora continua de cada tarea en particular.

En la representación por etapas, se establece una serie de niveles o etapas de madurez, con base en procesos probados, agrupados y ordenados. Y en sus relaciones entre procesos, estas etapas ayudan a constituir un mapa que ayuda a la organización a mejorar sus procesos. Dentro de cada etapa se observan varias tareas de un proceso, lo que beneficia a la organización en el enfoque de sus esfuerzos en tareas específicas para la mejora de los procesos. Cada tarea especifica diversas actividades que se describen en términos de prácticas. Éstas ayudan a la organización a realizar una implementación eficiente de cada tarea, lo que incrementa su grado de madurez, el cual avanza hasta que son cubiertos todos los objetivos de todas las tareas del proceso.

Ambas representaciones deben incluir metas tanto genéricas como específicas que serán medidas a través de los resultados de cada nivel de madurez, los cuales serán

apoyados por las prácticas que también deberán tener un enfoque tanto genérico como específico.

ISO 15504
(SPICE, Software Process Improvement and Capability of determination)

ISO/IEC 15504 es un nuevo estándar internacional para la evaluación, medición y mejora continua de los procesos de la ingeniería de *software*. Fundamenta su filosofía en el desarrollo de un conjunto de medidas de capacidad para cada uno de los procesos que conforman el ciclo de vida del *software*.

Objetivos principales del SPICE:

- Desarrollar un borrador de trabajo para un estándar de evaluación de procesos de *software*.
- Llevar a cabo los ensayos de la industria de la norma emergente.
- Promover la transferencia de tecnología de la evaluación de procesos de *software* a la industria del *software* a nivel mundial[53].

Características del proyecto SPICE[54]:

- Establece un marco y los requisitos para cualquier proceso de evaluación de procesos, así como los requisitos para los modelos de evaluación de los procesos.

[53] Norma ISO/IEC 15504. Sitio oficial de la Organización Internacional para la Estandarización (ISO). Disponible en http://goo.gl/calZuX. Recuperado: 08/07/2013.
[54] Fuente: Norma internacional ISO 9001. Sistema de gestión de calidad. Requisitos. Traducción disponible en http://goo.gl/GYzNK6. Recuperado: 08/07/2013.

- Proporciona los requisitos para cualquier modelo de evaluación de organizaciones.
- Ofrece guías para la definición de las competencias de un evaluador de procesos.
- Hoy día, tiene diez partes: de la 1 a la 7, completas; y de la 8 a la 10 en fase de desarrollo.
- Comprende evaluación de procesos, mejora de procesos, determinación de capacidad.
- En su parte 5, brinda un modelo de evaluación de procesos de ciclo de vida del *software* establecidos en el estándar ISO/IEC 12207, que define los procesos del ciclo de vida del desarrollo, mantenimiento y operación de los sistemas de *software*.
- En su parte 6, ofrece un modelo de evaluación de procesos para los procesos de ciclo de vida del sistema definidos en el estándar ISO/IEC 15288 que define los procesos del ciclo de vida del desarrollo, mantenimiento y operación de sistemas.
- En su parte 8, da un modelo de evaluación de procesos de servicios TIC, que serán puntualizados en el estándar ISO/IEC 20000-4, el cual definirá los procesos contenidos en la norma ISO/IEC 20000-1.
- Equivalencia y compatibilidad con CMMI. ISO forma parte del panel elaborador del modelo CMMI y SEI mantiene la compatibilidad y equivalencia de esta última con 15504. Sin embargo, CMMI-DEV aún no es un modelo conforme con esta norma (según lo requiere la norma ISO 15504 para todo modelo de evaluación de procesos).

4.6. Aplicación de la configuración del *software* en el desarrollo de sistemas

Al construir sistemas de información, es inevitable realizar cambios al sistema en cualquiera de sus fases del ciclo de vida, que pueden ser mínimos o muy significativos,

según la calidad de los requerimientos levantados en la fase de requerimientos. El impacto de un cambio de requerimientos en una fase avanzada del ciclo de vida de un sistema puede repercutir en el costo y tiempo de desarrollo, de aquí su importancia al recabarlos y documentarlos.

La gestión de configuración del *software* (GCS) es una actividad que ayuda a proteger el desarrollo del *software*, considerando que los cambios pueden ocurrir en cualquier fase del ciclo de vida. En este sentido, las actividades de CGS ayudan a identificar los cambios, controlarlos, implementarlos de forma adecuada y darlos a conocer al equipo de desarrollo.

4.6.1. Importancia

Cuando se crean sistemas o *software* de forma profesional, el resultado del proceso de desarrollo debe considerarse en tres partes principales:

1. *Programa ejecutable*. Sistema terminado tanto en su código como en su compilación para generar el ejecutable.
2. *Documentación*. Manuales técnicos y de usuario donde se detalla la arquitectura del *software* y sus componentes, así como su manera de operar.
3. *Datos.* Tipo de información que se almacena, procesa y que entrega el *software* en cuestión.

El conjunto de estos tres elementos se denomina “configuración del *software*”. Conforme se avanza en el ciclo de vida del *software*, independientemente del modelo empleado, estos componentes serán más extensos y complejos. Cada requerimiento lleva a una especificación; a su vez, cada especificación conduce a un proceso de implementación dentro del proyecto, y así sucesivamente, por lo que la configuración del *software* va evolucionando y cambiando según se avanza en el proyecto. Por eso, la

carencia de una gestión adecuada que documente, vigile y controle los cambios, puede impactar grandemente en el desarrollo del sistema en fases posteriores, lo que derivaría en un rediseño del proceso y el empleo de más recursos y tiempo.

4.6.2. Rol

El rol de la gestión de la configuración del *software* es crucial para prevenir que el *software* resultante sea de mala calidad. Por ello esta tarea debe ser efectuada a través de todas las fases del ciclo de vida, ya sea por una persona especialmente dedicada a monitorear los cambios, o por los miembros del equipo de desarrollo involucrado en cada fase.

4.6.3. Actividades

Para controlar los cambios en la configuración, es importante realizar lo siguiente[55]:

1. *Analizar el problema y especificar el cambio.* Comienza con la identificación de algún problema en algún requerimiento o con la propuesta por parte del cliente o algún desarrollador para el cambio en un requerimiento. La propuesta o problema son analizados para verificar su validez; luego, se entrega el resultado del análisis a quien identificó el problema o solicitó el cambio.

2. *Análisis del cambio y análisis de costos.* Los efectos que puede tener el cambio propuesto son evaluados empleando la información de rastreo y el conocimiento general de los requerimientos del sistema. El costo del cambio se determina en términos de la modificación del documento de

[55] Pressman, R. *Ingeniería de* software, 5.ª ed., p. 156.

requerimientos y con base en el impacto en el diseño e implementación del sistema. Realizado el análisis, se debe tomar una decisión sobre si se continuará o no con el cambio de requerimiento.

3. *Implementación del cambio.* Una vez que se decide continuar con el cambio en el requerimiento, éste debe efectuarse en el documento de requerimientos y, de ser necesario, en el diseño e implementación del sistema. El documento de requerimientos se organizará de tal forma que sea posible hacer cambios en su contenido sin tener que redactar grandes partes de él. Los cambios en el documento se realizarán minimizando las referencias externas y haciendo el documento tan modular como sea posible para permitir la modificación de alguna sección sin afectar otras partes del documento.

RESUMEN DE LA UNIDAD

El desarrollo es una parte del ciclo de vida de los sistemas de información. Independientemente del modelo que se siga, durante esta fase se convierten los requerimientos analizados y validados de la fase de análisis, además de la documentación derivada de la fase de diseño, en un producto funcional. Algunas metodologías de desarrollo se enfocan a la entrega de un producto final (como el modelo en cascada), o a pequeños prototipos o programas funcionales que van siendo evaluados conforme se entregan a los clientes. En todo caso, el objetivo es la construcción del sistema en sí, para luego evaluarlo en la fase de pruebas.

En el ciclo de vida de un sistema de información, es necesario establecer los roles que cada participante tendrá durante todo el proceso, comenzando con la asignación de roles administrativos que serán los encargados de la gestión de recursos, hasta la asignación de los diversos roles técnicos donde se encargará a cada persona una tarea específicamente enfocada en cierto proceso del ciclo de vida, pasando por los programadores, diseñadores, *testers*, etcétera.

Es importante el seguimiento de una metodología, sobre todo si se busca la generación de sistemas de calidad. Con este propósito, algunas metodologías actuales pueden respaldarse en estándares para el desarrollo de *software* emitidas por entidades como la ISO o el Instituto de Ingeniería de Software, que se han encargado de encuadrar lo que podemos denominar “buenas prácticas” en el desarrollo de *software*, en una serie de recomendaciones y documentos que conforman estándares actuales de desarrollo, para la creación de *software* de calidad.

GLOSARIO DE LA UNIDAD

Método

Del griego *meta,* ‘más allá’; y *hodos* ‘camino’, se refiere al modo de decir o hacer algo con orden. Es la manera de conducir el pensamiento y las acciones para alcanzar una meta preestablecida. Incluye diversas técnicas y procedimientos adecuados al objeto a tratar.

Ingeniería del *software*

Aplicación de procesos sistemáticos y disciplinados para el desarrollo, operación y mantenimiento de *software*.

Prototipo

Versión preliminar de un sistema que sirve de modelo para fases posteriores.

Requerimiento o requisito

Condición o facultad que necesita un usuario para resolver un problema; o que debe poseer un sistema o componente de un sistema para satisfacer una especificación, estándar, condición de contrato u otra formalidad impuesta documentalmente. También se entiende como el documento donde se reúnen los puntos anteriores.

Técnica

Del griego *téchne*, ‘arte’, ‘cómo hacer algo’. Es un conjunto de saberes prácticos o procedimientos para obtener resultados deseados. Supone que, en situaciones similares, repetir conductas o llevar a cabo un mismo procedimiento producirá el mismo efecto. Por tanto, se trata de una forma de actuar ordenada que consiste en la repetición sistemática de ciertas

acciones; y exige destrezas intelectuales y manuales, así como herramientas y el conocimiento para manipularlas.

Validación

Confirmación, mediante examen y aportación de pruebas objetivas, de que se cumplen los requisitos concretos para un uso determinado. Responde a la pregunta *¿estamos construyendo el producto correcto?*

ACTIVIDADES DE APRENDIZAJE

ACTIVIDAD 1

Elabora un árbol jerárquico de los roles más comunes en el desarrollo de sistemas de información. Incluye un comentario sobre dicha estructura.

ACTIVIDAD 2

Discute con tus compañeros sobre los roles en el proceso de desarrollo de sistemas de información. Haz un comentario sobre la importancia de establecer roles en el proceso de desarrollo de un sistema de información y la forma como éstos pueden relacionarse.

A manera de retroalimentación, comenta el aporte de dos de tus compañeros.

ACTIVIDAD 3

Elabora un cuadro sinóptico donde describas las características de los ciclos de vida convencionales y los actuales. Luego, realiza un comentario sobre los elementos que integran ambos modelos.

ACTIVIDAD 4

Elabora un cuadro comparativo de los diversos modelos para el aseguramiento de la calidad del *software*, donde destaques sus características, vigencia y uso.

ACTIVIDAD 5

En el foro de la asignatura, discute con tus compañeros las siguientes preguntas:

a. ¿Qué es la configuración del sistema?
b. ¿Por qué es importante considerar la configuración del sistema en la fase de desarrollo de sistemas?
c. Menciona dos actividades que implica la configuración del sistema.

CUESTIONARIO DE REFORZAMIENTO

Responde las siguientes preguntas.

1. ¿Qué es la fase de diseño de sistemas?
2. ¿Qué es un rol?
3. Menciona tres ejemplos de roles y explícalos brevemente.
4. ¿Qué es el ciclo de vida de los sistemas?
5. ¿Cuáles son las diferencias y similitudes principales entre los modelos de ciclo de vida tradicionales y los actuales?
6. ¿Qué es la calidad del *software*?
7. Explica brevemente algún modelo para garantizar la calidad del *software.*
8. ¿Qué es la configuración del sistema?
9. Menciona tres funciones realizadas por la configuración del sistema.
10. ¿Qué es el control de cambios de la configuración y cuál es su importancia?

LO QUE APRENDÍ

Elabora un cuadro sinóptico donde describas el proceso de diseño de sistemas, sus elementos integrales y su ubicación en los diversos modelos de ciclo de vida de sistemas.

EXAMEN DE AUTOEVALUACIÓN

Relaciona ambas columnas.

1. Enfoque de diseño de sistemas donde el sistema presenta pasos secuenciales y lineales que van desde el análisis de la información hasta su puesta en marcha y mantenimiento. __ _____	a. PSP
2. Se basa en desarrollar un modelo inicial del sistema que pone a consideración de los clientes para enriquecer su funcionamiento a través de los comentarios que éstos realicen. _____	b. Calidad del *software*
3. Metodología donde todas las tareas y actividades que el ingeniero de *software* debe realizar durante el proceso de desarrollo de un producto de *software* están puntualmente definidas en un conjunto de documentos conocidos como *scripts*. _____	c. Ingeniería de *software*
4. Metodología para dirigir el trabajo de mejora y desarrollo de *software*, y establecer un entorno donde el trabajo efectivo de equipo sea normal y natural. _____	d. TSP
5. Concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo directamente documentados, y con las características implícitas que se espera de todo *software* desarrollado de manera profesional. _____	e. Modelo en cascada

6. Patrón de acciones planificado y sistemático que se requiere para asegurar la calidad del *software*. ___	f. CMMI
7. Marco de referencia de la capacidad de las organizaciones de desarrollo de *software* en el desempeño de sus diferentes procesos que proporciona una base para evaluar su madurez y una guía para implementar una estrategia para su mejora continua. _____	g. Modelo de prototipos
8. Establecimiento y uso de principios robustos de la ingeniería a fin de obtener económicamente *software* que sea fiable y funcione de manera eficiente sobre máquinas reales. _____	h. SPICE
9. Nuevo estándar internacional para la evaluación, medición y mejora continua de los procesos de la ingeniería de *software*. Basa su filosofía en el desarrollo de un conjunto de medidas de capacidad para cada uno de los procesos que conforman el ciclo de vida del *software*. _____	i. GCS
10. Su gestión consiste en una actividad que ayuda a proteger el desarrollo del *software*, considerando, especialmente, que los cambios pueden ocurrir en cualquier fase del ciclo de vida. ____	j. SQA

MESOGRAFÍA

BIBLIOGRAFÍA RECOMENDADA

Autor	Capítulo	Páginas
Roger S. Pressman, *Ingeniería del* software*. Un enfoque práctico*	10	132-165
Jonás Montilva C. Judith Barrios A. *Desarrollo de* software *empresarial.*	3	18-29

BIBLIOGRAFÍA BÁSICA (REFERENCIAS)

- Pressman, Ian. *Ingeniería del* software. *Un enfoque práctico,* 5.ª ed. España: McGraw-Hill, 2002, 601 pp.
- Montilva y Barrios. *Desarrollo de* software *empresarial*. Universidad de los Andes, 2007, 121 pp.
- Vila, Ruiz, y Ramos. *Modelos de evaluación y mejora de procesos: análisis comparativo*. España, 2005, 18 pp.
- Cortes. *Introducción al análisis de sistemas y la ingeniería de* software. Costa Rica: Euned, 2006, 168 pp.
- Somerville, Ian. *Ingeniería del* software, 7.ª ed. España, 2005. 691 pp.

BIBLIOGRAFÍA COMPLEMENTARIA (TEMARIO ANALÍTICO)

1. Bardou, Louis. *Mantenimiento y soporte logístico de los sistemas informáticos,* México: Alfa Omega-Marcombo, 2004, 292 pp.
2. Bochinno, William A. *Sistemas de información para la administración, técnicas e instrumentos.* México: Trillas, 2002.
3. Bonsón, Enrique. *Tecnologías inteligentes para la gestión empresarial*. México: Alfa Omega-Rama, 2002, 258 pp.
4. Cornella, Alfons. *Información digital para la empresa, una introducción a los servicios de información electrónica.* México: Alfa Omega-Marcombo, 2004, 196 pp.
5. Lardent, Alberto R. *Sistemas de información para la gestión empresarial, procedimientos, seguridad y auditoría*. Buenos Aires: Pearson Education-Prentice Hall, 2001.
6. Levine, Guillermo. *Computación y programación moderna*. México: Addison Wesley, 2000.
7. Long, Nancy y Larry Long. *Introducción a las computadoras y a los sistemas de información*, *edición Internet.* México: Prentice Hall, 1999, 416 pp.
8. McLeod, Raymond, Jr. *Sistemas de información gerencial,* 7.ª ed. México: Prentice Hall, 1999, 688 pp.
9. Oz, Effy. *Administración de sistemas de información*, 2.ª ed. México: Thomson-Learning, 2001, 578 pp.
10. Peña R., Baeza-Yates, R. y Rodríguez, J. *Gestión digital de la información. de bits a bibliotecas digitales y la web.* México: Alfa Omega-Rama, 2004, 464 pp.
11. Piattini, Mario; J. Antonio Calvo-Manzano, Joaquín Cervera y Luis Fernández. *Análisis y diseño detallado de aplicaciones informáticas de gestión.* México: Alfa Omega-Rama, 2004, 728 pp.
12. Stair, Ralph M. *Principios de sistemas de información*, 4.ª ed. México: Thomson Learning, 2002, 692 pp.

13. Walker, D. W. *Sistemas e información para la administración.* México: Alfa Omega-Marcombo, 2001, 360 pp.

SITIOS ELECTRÓNICOS

Dirección electrónica	Descripción
http://goo.gl/AQO79R	El aseguramiento de la calidad del *software*, por el departamento de informática de la Universidad Técnica de Federico Santa María, Colombia.
http://jmpovedar.files.wordpress.com/2011/08/modelos-de-desarrollo-del-software.pptx.	Presentación electrónica sobre modelos de desarrollo de *software*, por José Manuel Poveda.
http://goo.gl/FNsJb6	El desarrollo del *software*, por Fernando Sáez Vaca, Fundación Rogelio Segovia para el Desarrollo de las Telecomunicaciones.
http://goo.gl/x3ErvT	Apuntes de Taller de Ingeniería de Software, David Fuller Padilla, 2003.

UNIDAD 5
IMPLEMENTACIÓN DE SISTEMAS

OBJETIVO ESPECÍFICO

Al finalizar la unidad, el alumno identificará los conceptos relacionados a la fase de implementación de sistemas en el ciclo de vida de los sistemas de información.

INTRODUCCIÓN

La fase de implementación de un sistema puede considerarse como la puesta en marcha del mismo. Debe incluir varios elementos funcionales para que el sistema trabaje de manera correcta, desde el *hardware* (elementos físicos para poder instalar el sistema), *software* (sistemas operativos, manejador de base de datos, lenguajes de programación, etcétera), hasta la descripción de los procesos que deben ser realizados para poder efectuar una implementación exitosa.

En esta unidad, abordaremos los aspectos necesarios para una correcta implementación de un sistema: requisitos de implementación que detallan los elementos de *hardware* y *software* indispensables para el correcto funcionamiento del sistema, forma de almacenamiento de los datos procesados y la manera de implementación de cada proceso que compone al sistema de información desarrollado.

LO QUE SÉ

Elabora una ficha de texto donde escribas tu concepto de la fase de implementación de un sistema y sus principales componentes.

TEMARIO DETALLADO (8 HORAS)

5.1. Definición de implementación de sistemas
5.2. Elemento personal
5.3. Elemento *hardware*
5.4. Elemento *software*
    5.4.1. Sistema operativo
    5.4.2. Sistemas de aplicación
    5.4.3. Utilidades del desarrollo del sistema
    5. 4.4. Medios útiles de mantenimiento del sistema
5.5. Almacenamiento de datos
5.6. Implementación de procesos
5.7. Requisitos de una implementación
5.8. Representación de una implementación

5.1. Definición de implementación de sistemas[56]

Se entiende como implementación de sistemas todas las actividades necesarias para convertir el sistema anterior al nuevo sistema; o bien el proceso que asegura la operatividad del sistema de información y que permite al usuario obtener beneficios por su operación.

Todo sistema de información debe cumplir con su ciclo de vida; y en este ciclo, al llegar a su madurez, el sistema de información puede presentar dos escenarios: aún ser de utilidad y es posible alargar su vida a través de una actualización; o ya no responder a las necesidades de la empresa y, por tanto, es necesario uno nuevo para sustituirlo. En este último caso, la implementación de un sistema nuevo, exige la modificación de los procesos internos de la empresa a los cuales estará asociado, comenzando por concientizar a los usuarios que tendrán contacto con él sobre los beneficios que implicará para su trabajo. En todo caso, la implementación de un sistema de información es un proceso que debe ser planificado de manera minuciosa para permitir una instalación y puesta en marcha exitosa.

5.2. Elemento personal

Es necesario comenzar a concientizar a las personas sobre la importancia del nuevo sistema y de los beneficios que tendrá para la empresa y el trabajo de los usuarios. Además, cuando un sistema de información está por ser implementado se debe capacitar al personal que lo usará; incluso en algunas ocasiones habrá de reasignarse al personal dentro de la empresa para cubrir con los requisitos operativos del sistema y lograr que se cumplan los objetivos para el cual fue diseñado.

[56] Apuntes de Ingeniería de *software*. Instituto de Estudios Superiores de Campeche. Disponible en http://goo.gl/MFY6h0. Recuperado: 15/10/2013.

Para establecer un programa de capacitación efectiva, es importante identificar a dos tipos de personal asociados al sistema de información: personal técnico (ingenieros de *software*, analistas, programadores, etcétera) y usuarios finales (personal operativo de la empresa que estará en contacto cotidiano con el sistema). El primero debe recibir entrenamiento enfocado a las herramientas de *hardware* y *software* asociadas al sistema de información; por ejemplo, familiarización con el lenguaje de programación, operación y administración del sistema manejador de base de datos, etcétera. Y los usuarios finales del sistema serán entrenados en los aspectos de operación del mismo, como el empleo de los equipos donde será instalado, identificación de problemas, uso del sistema y sus interfaces, ingreso de datos, impresión de reportes, etcétera.

En cuanto a la capacitación, puede ser impartida ya sea por la misma empresa o por terceros, según el programa de capacitación diseñado por la empresa acorde con sus necesidades. En este orden, desde el punto de vista gerencial, hay tres posturas dentro de la empresa:

- *Gerencias adversas a la capacitación.* Se presenta cuando los gerentes de la empresa no están en favor de la impartición de cursos de capacitación. Basan sus argumentos en que resulta costoso y en que el personal una vez capacitado prefiere dejar la empresa optando por un mejor puesto en otra.

- *Gerencias promotoras de la capacitación.* Este tipo de gerencias se presenta cuando los directivos están conscientes de las ventajas que se adquieren al tener personal calificado y actualizado. Motivan a sus empleados a permanecer actualizados en todo momento y establecen planes de capacitación acordes con las necesidades.

- *Gerencias indiferentes*. Dentro de este tipo de gerencias, los directivos no están ni a favor ni en contra de la capacitación, aunque no la perciben como una

necesidad. Este tipo de directivos aprobarán regularmente cursos de capacitación siempre y cuando se encuentren justificados.

Reasignación del personal

En la mayoría de los casos, cuando es implementado un nuevo sistema de información, será necesaria la reasignación de personal a otras áreas de la empresa, ya sea para capacitarlo en el manejo del sistema, por modificaciones en los procesos internos de la empresa asociados al sistema, reasignación de puestos de trabajo o reducción de los mismos, entre otros motivos. La reasignación de personal no debe tomarse a la ligera; al contrario, será considerada en el plan de implementación del sistema para tomar las previsiones necesarias y manejarla de forma adecuada.

Procedimientos y métodos

Cuando se implementa un sistema de información, tendrá impacto en los métodos o procedimientos de la empresa, cambiará la manera como se realizaban algunas tareas y, por tanto, los procesos asociados a éstas. Por ejemplo, en los bancos, antes del auge de las tecnologías de la información, las operaciones bancarias tenían que ser realizadas en su totalidad directamente en la sucursal, revisiones de saldo, depósitos y retiros de dinero, cobro de cheques, etcétera. Hoy día, con los nuevos sistemas interbancarios, es posible hacer la mayoría de las operaciones desde nuestras casas.

Seguridad física y de datos

Cuando se habla de seguridad desde el punto de vista informático, no solamente se hace referencia a la protección de la información que procesan los sistemas, sino también al aspecto físico; es decir, tanto el *hardware* como el *software*. En el primer caso, se debe considerar que el lugar físico donde será alojado el servidor que contendrá al nuevo sistema sea el adecuado: tenga un control de acceso de personal, esté perfectamente ventilado con instalaciones eléctricas y de comunicaciones adecuadas, etcétera. En lo que a *software* se refiere, se habrá de verificar que se cuente con los programas necesarios que darán soporte al sistema de información en su operación, por ejemplo, un sistema operativo multiusuario robusto (preferentemente de red), lenguaje de programación del mismo sistema, manejador de base de datos, antivirus, cortafuegos, etcétera.

Adicionalmente, es necesario llevar una administración de los recursos del sistema de forma adecuada, monitoreando y creando cuentas de usuarios, actualizando el sistema operativo y el antivirus, realizando copias de seguridad de los datos, etcétera.

Documentación

La implementación del sistema se acompañará siempre de la documentación necesaria para entender el funcionamiento del sistema y su operación. Lo anterior en dos tipos de documentos, un manual de usuario donde se describa la forma de instalar, operar y desinstalar el sistema de forma correcta; y un manual técnico, donde se concentre toda la documentación generada a lo largo del proceso de desarrollo del mismo sistema, como diagramas del sistema, diagramas de flujo, entidad relación, definición de datos, código fuente, etcétera.

Un sistema bien documentado ofrece las siguientes ventajas:

- *Mejora la comunicación.* Aumenta el nivel de entendimiento en los usuarios de todos los niveles y facilita la comprensión del sistema.

- *Control de avance de proyectos*. Cada etapa del desarrollo de un sistema genera documentación mediante la cual es posible evaluar si los requerimientos de un cierto módulo o un conjunto de ellos fueron implementados de forma correcta; si el presupuesto se ha empleado de forma adecuada y, lo más importante, si el sistema en sí cumplirá con los objetivos para el cual fue construido.

- *Referencia histórica.* La documentación técnica permite conocer la estructura de un sistema y su funcionamiento, lo que a futuro permitirá que se le realicen mejoras, un buen mantenimiento, cambios en su funcionalidad y, cuando el tiempo lo indique, sea suplantado por uno nuevo.

- *Enseñanza o capacitación.* Los manuales de usuario y técnico son la base para los programas de capacitación asociados al sistema de información; permiten que los usuarios conozcan y manejen de forma adecuada el sistema, y el personal técnico domine su estructura para establecer planes de contingencia, mantenimiento y planeación de futuros sistemas.

- *Independencia.* Una buena documentación técnica permitirá a la empresa prescindir en un futuro de los desarrolladores originales del sistema, y dará pie a que nuevos desarrolladores realicen el mantenimiento y las mejoras.

5.3. Elemento *hardware*

Por lo general, cuando se implementa un nuevo sistema de información, viene asociado con la necesidad de adquirir o adaptar el *hardware* existente a las exigencias del sistema para su correcto funcionamiento. Todo *software*, sin excepción, tiene una serie de requerimientos mínimos de *hardware*; por ejemplo, un video juego solicitará un equipo con una cantidad mínima de memoria RAM, espacio en disco duro, tarjeta gráfica con capacidad de gráficos en 3D acelerados, etcétera.

En general, al momento de diseñar el sistema se deben ir determinando las características físicas del equipo (un servidor) donde será instalada la aplicación principal, y a la par las de los equipos, que serán los clientes de la aplicación:

- Tipo de procesador y velocidad
- Memoria RAM
- Capacidad y cantidad de discos duros
- Dispositivos de interconectividad a redes como tarjetas de red, módems, ruteadores, puertas de enlace, etcétera
- Monitor
- Teclado

En lo referente al entorno físico donde será colocado el servidor, debe reunir estas características:

- Controles de acceso de personal al cuarto del servidor
- Instalaciones eléctricas adecuadas
- Extintores
- Cámaras de vigilancia
- Control de clima (ventiladores o aire acondicionado)

- Si la empresa no cuenta con una red de datos, deberá considerar el diseño e instalación de ésta.

5.4. Elemento *software*

Los sistemas de información son un *software* altamente especializado y, por así dominarlo, personalizado, por lo que al instalarlo se deben cumplir requisitos especificados en su diseño. En este orden, para establecer de manera adecuada los requisitos de *software*, es deseable generar un documento que permita analizar su entorno y funcionalidad para determinar. Este documento puede contener lo siguiente:

- Una descripción funcional detallada
- Una indicación de los requisitos de rendimiento
- Restricciones para el diseño
- Criterios de validación

En lo que se refiere al formato de la especificación de requisitos del *software*, puede tener esta estructura[57]:

- Panorama del producto y resumen
- Ambientes de desarrollo/operación/mantenimiento
- Interfaces externas y flujo de datos
- Despliegues al usuario/formato de informes
- Resumen de comandos del usuario
- Diagrama de flujo de datos de alto nivel
- Fuentes y destinos lógicos de datos

[57]Basado en Especiación de Requisitos según el estándar de IEEE830. Disponible en http://goo.gl/8tns6a. Recuperado: 15/10/2013.
Para mayor información, en los Apuntes de la asignatura Informática II. SUAyED, FCA, UNAM, 2012.

- Almacenamiento lógico de datos
- Diccionario lógico de datos
- Especificaciones funcionales
- Requisitos de operación
- Condiciones de excepción/manejo de excepciones
- Subconjuntos iniciales y prioridades de codificación
- Modificaciones y mejoras previstas
- Criterios de aceptación
- Pruebas funcionales y de operación
- Estándares de documentación
- Guías de diseño
- Fuentes de información
- Glosario de términos

5.4.1. Sistema operativo

El principal elemento a considerar dentro de los requisitos de *software* de un sistema es el sistema operativo con el que trabajará (el sistema operativo administra los recursos de una computadora y es el que permite que se ejecuten los otros programas sobre él). De modo general, cuando se diseña un sistema, lo ideal es pensar en que éste pueda ser instalado en diversos tipos de sistemas operativos, aunque no siempre es el caso. Así, determinar el sistema operativo adecuado para el sistema de información no sólo es esencial, sino que tendrá impacto en desarrollos futuros o mejoras del mismo sistema, ya que permitirá determinar su escalabilidad y compatibilidad con otras plataformas y nuevas versiones del mismo sistema operativo.

Los desarrolladores de *software* tomarán en cuenta, además del tipo de sistema, las diversas versiones de uno solo. Por ejemplo, si es de 32 o 64 bits, si es sistema

operativo para estación de trabajo o servidor, si es compatible el cliente en sistemas operativos de dispositivos móviles, etcétera.

A continuación, se muestra un cuadro donde se presentan los principales sistemas operativos en el mercado.

Servidores	Usuarios finales	Móviles
Debian GNU/Linux	Distribuciones Linux	Android
FreeBSD	Mac OS	iOS
Mac OS X	Windows	Blackberry
NetBSD		Windows Phone
NetMax Professional		Bada
Novell NetWare		Firefox OS
OpenBSD		Ubuntu Mobile
Red Hat Enterprise Linux ES		Symbian OS
Red Hat Linux		Sistemas propietarios
Sun Solaris		
SUSE Linux Enterprise Server		
Ubuntu		
Windows 2000 Server		
Windows ME		
Windows NT Server		
Windows Server 2003		
Windows Server 2008		
Windows Server 2012		

Tabla 5.1. Sistemas operativos modernos.

5.4.2. Sistemas de aplicación

El *software* de aplicación, o aplicaciones, son programas que permiten al usuario trabajar con las computadoras. Por lo general, es *software* que podemos denominar “de uso común”, como procesadores de texto, hojas de cálculo, etcétera. La importancia de considerar estas aplicaciones dentro del plan de implementación de un sistema de información radica en que, través de ellas, se generarán los datos de entrada del sistema y se obtendrán los reportes de salida del mismo. En la mayoría de los casos, los datos de entrada pueden contenerse dentro de un archivo de texto plano (archivo .txt) o en una hoja de cálculo (archivos .xls); mientras que los datos de salida se guardan en un reporte compatible con ciertos procesadores de texto o en las mismas hojas de cálculo, según las especificaciones establecidas en el diseño del sistema.

Algunos ejemplos de *software* aplicaciones:

- Procesadores de texto (Bloc de Notas)
- Editores (PhotoShop para el Diseño Gráfico)
- Hojas de cálculo (MS Excel)
- Sistemas gestores de bases de datos (MySQL)
- Programas de comunicaciones (MSN Messenger)
- Paquetes integrados (Word, Excel, PowerPoint…)
- Programas de diseño asistido por computador (AutoCAD)

5.4.3. Utilidades de desarrollo del sistema

Las utilidades o utilerías de desarrollo para sistemas de información son aquellas herramientas de *software* que serán empleadas en la construcción del sistema, todas acordes con los requisitos funcionales y las características técnicas del sistema. Su

objetivo principal es aumentar la productividad de los desarrolladores, así como asistir y acelerar en el proceso de desarrollo del sistema.

Algunas utilerías:

- Lenguajes de programación
- Administradores gráficos de bases de datos
- Programas para diagramación

Ventajas del empleo de utilerías en el desarrollo de sistemas:

- Minimizar la curva de aprendizaje
- Proponer patrones de usabilidad
- Generar interfaces de usuario
- Generar código
- Flexibilidad de configuración
- Adaptabilidad

5.4.4. Medios útiles de mantenimiento del sistema

Un sistema saludable es al que se le puede realizar un mantenimiento simple sin que afecte a los usuarios que dependen de él. En este sentido, para facilitar el mantenimiento, son de ayuda las técnicas siguientes:

1. Escribir el programa modular, así sólo será necesario alterar un módulo en particular.

2. Incluir documentación adecuada y útil, comentando lo que no es obvio: para qué es la rutina, parámetros requeridos y zonas críticas.

3. Utilizar identificadores cuyos nombres sean significativos y se relacionen con el problema a resolver.

4. Simplificar al máximo las expresiones booleanas utilizadas en ciclos y selecciones.

5.5. Almacenamiento de datos

Uno de los aspectos principales de los sistemas de información es su base de datos, el sitio donde serán almacenados los datos para su empleo dentro del sistema. A lo largo de la historia de la ingeniería de *software*, el almacenamiento de datos siempre ha estado presente, y así como las técnicas de ingeniería han ido evolucionando, también las bases de datos lo han hecho para tener sistemas más robustos y eficientes.

La selección del sistema manejador de base de datos adecuado es esencial para el desarrollo del sistema, ya que no solamente contendrá la base de datos que estará asociada al sistema, sino que adicionalmente deberá permitir a los desarrolladores crear objetos de manipulación de datos más complejos como funciones, procedimientos almacenados, *triggers*, etcétera, que facilitarán el procesamiento de la información. Asimismo, este manejador de base de datos permitirá administrar las características de seguridad asociadas directamente a la base, como la administración de usuarios de la base, permisos, roles y creación de respaldos, entre otros.

Hoy, existen diferentes opciones de manejadores de base de datos en el mercado, ya sea de código abierto o bajo licencia:

- *SQL Server.* Manejador de base de datos desarrollado por Microsoft para trabajar sobre sus sistemas operativos de la familia Windows.

- *ORACLE.* Manejador de base de datos empresarial desarrollado para trabajar con sistemas operativos de la familia UNIX.
- *Postgresql.* Manejador de base de datos gratuito de código libre para desarrollo de aplicaciones personales que trabajo con sistemas operativos Linux y Windows.

5.6. Implementación de procesos

La implementación consiste en establecer el sistema en términos de componentes, es decir, los archivos de código fuente, de cabecera, ejecutables, etcétera. Tiene como fines planificar las integraciones a realizar en cada interacción (en este punto, produce un sistema que se implementa en pequeños pasos), distribuir los componentes ejecutables, ejecutar las tareas de cada clase, así como los subsistemas encontrados durante el diseño por medio de componentes de archivos fuente, y probar los componentes individualmente e integrarlos con el sistema.

5.7. Requisitos de una implementación

Los requisitos para la implantación de un sistema dependen de muchos factores, como las capacidades del *hardware*, la topología de la implementación del servidor, las características de la solución (por ejemplo, si tiene particiones distribuidas en varios servidores), los contratos de nivel de servicio y la complejidad del modelo de datos, etcétera. Pero en esencia parte de seguir los lineamientos establecidos en la fase de diseño, hasta completar todo el ciclo de desarrollo de *software*.

5.8. Representación de una implementación

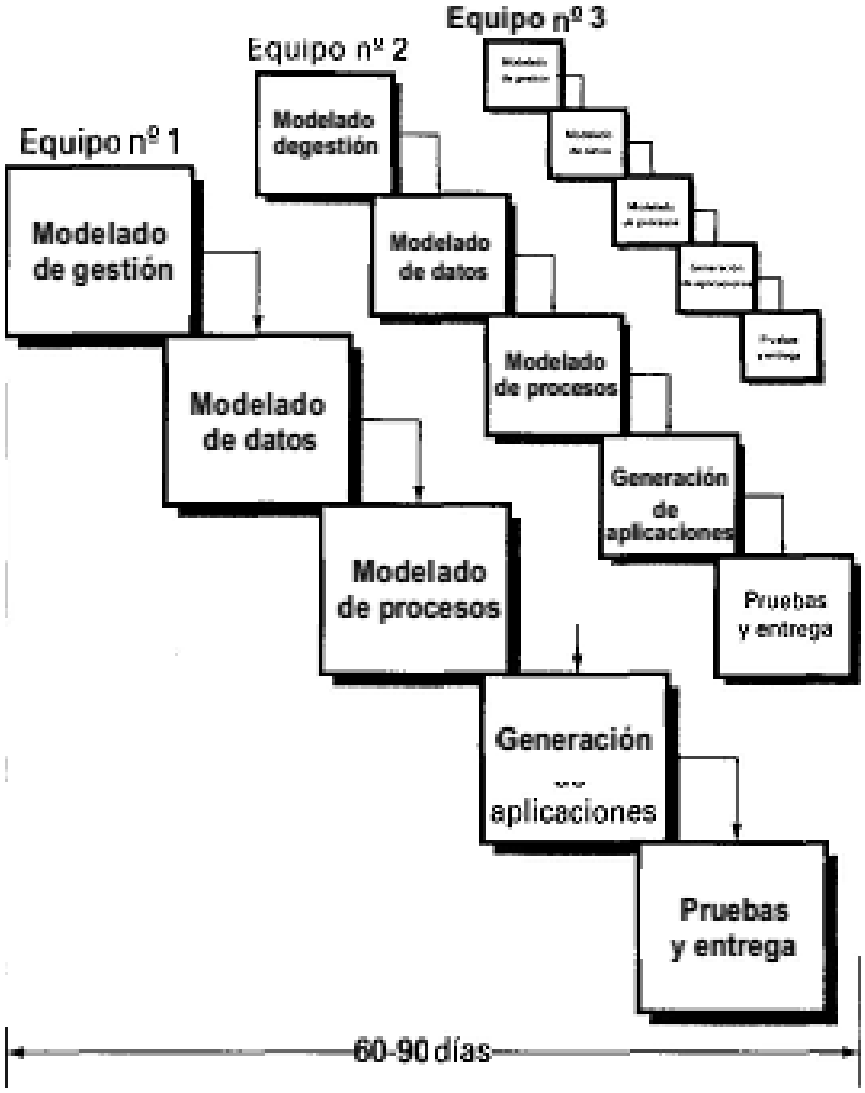


Figura 5.3 Modelo DRA.[58]

Tomando como ejemplo el modelo de desarrollo rápido de aplicaciones (DRA), podemos generar una representación gráfica de la implementación de un sistema, donde las limitaciones de tiempo impuestas demanden “ámbito en escalas”. Si una aplicación de gestión puede modularse de forma que permita completar cada una de las funciones principales en menos de tres meses (utilizando el enfoque descrito

[58] Pressman, R. *Ingeniería del* software. *Un enfoque práctico*, 5.ª ed., p. 22.

anteriormente), de esta manera cada una de las funciones pueden ser afrontadas por un equipo DRA separado y ser integradas en un solo conjunto.

RESUMEN DE LA UNIDAD

La implementación de un sistema de información es la fase donde el sistema se pondrá al alcance de los usuarios finales. En este momento, se instala el sistema y los usuarios tendrán acceso a él; pero antes debe elaborarse un plan de implementación. Éste se realiza a lo largo del proceso de desarrollo del sistema, donde podrán definirse aspectos críticos como las características físicas tanto del servidor donde se encontrará la base de datos y parte principal del sistema, como de las estaciones de trabajo (clientes) que se conectarán a él, y la estructura física donde se encontrará el servidor.

Otro aspecto a definirse son las características del *software* que acompañará al sistema, comenzando con el mismo sistema operativo, donde el sistema será ejecutado y se administrará parte de la seguridad. De manera adicional, se definirán el manejador de base datos, las aplicaciones que ayudarán a un mejor funcionamiento del sistema y a su mantenimiento, y las aplicaciones convencionales como hojas de cálculo, procesadores de texto, etcétera, que permitirán a los usuarios alimentar los datos de entrada del sistema e interpretar los reportes de datos de salida.

Por último, la implementación también debe considerar el factor humano, llevando a la práctica planes de capacitación para los diversos perfiles de personal que interactuarán con el sistema, ya sea de forma técnica o convencional.

GLOSARIO DE LA UNIDAD

Herramientas de *software*

Programas creados con la intención de ayudar a los usuarios en la realización de diversas tareas, como programación, creación de sitios web, administración, etcétera.

Manejador de base de datos (DBMS)

Programa de computadora que permite la creación, administración y manipulación de varias bases de datos.

Sistema operativo

Software que permite a los usuarios interactuar con los equipos de cómputo, realiza la administración de los recursos de la computadora y hace que los diferentes programas sean ejecutados en su entorno.

Utilerías o utilidades

Mismo caso que las herramientas de *software*.

ACTIVIDADES DE APRENDIZAJE

ACTIVIDAD 1

Discute con tus compañeros sobre el impacto de un nuevo sistema de información en el personal de una empresa, con base en las siguientes preguntas.

1. ¿Por qué es indispensable capacitar al personal al instalar un nuevo sistema de información?
2. ¿Por qué es necesaria la reasignación de puestos laborales al instalar un nuevo sistema?
3. ¿Cuál es principal problema al que se enfrentan tanto desarrolladores como gerentes y usuarios al implementar un nuevo sistema de información?

ACTIVIDAD 2

Realiza una investigación sobre tres ejemplos de requisitos de *software* y *hardware* en la implementación de sistemas de información. Y con base en esa investigación haz un cuadro comparativo con los elementos de *hardware* y *software* indicados en cada uno.

ACTIVIDAD 3

Realiza una investigación sobre tres herramientas o utilidades auxiliares en el desarrollo de sistemas de información. Y con base en esa investigación haz una síntesis donde describas cada herramienta y su utilidad.

ACTIVIDAD 4

Discute con tus compañeros sobre la selección de *software* para la implementación de un sistema de información, con base en las siguientes preguntas.

1. ¿Por qué es importante especificar tanto el *software* operativo (sistemas operativos, DBMS, etcétera) como el soporte (utilidades, lenguajes de programación, etcétera) antes de realizar la implementación de un sistema?
2. ¿Por qué son necesarias las aplicaciones al implementar un sistema de información?
3. ¿Cómo determinarías el *software* adecuado para la implementación de un sistema de información?

CUESTIONARIO DE REFORZAMIENTO

Responde las siguientes preguntas.

1. ¿Qué es la implementación de un sistema?
2. ¿Qué aspectos sobre el personal se deben considerar al implementar un sistema?
3. ¿Qué es la capacitación?
4. ¿Cuáles son los enfoques que puede tomar la gerencia con respecto a la capacitación?
5. ¿Qué son las aplicaciones y cómo se relacionan a los sistemas?
6. ¿Qué son las utilerías y cómo ayudan en el desarrollo de un sistema?
7. ¿Por qué es importante la selección del sistema operativo en la implementación de un sistema de información?
8. ¿Qué aspectos hay que considerar en cuanto a la selección de un DBMS para un sistema de información?
9. ¿Cómo impacta a los procesos de la empresa la implementación de un sistema de información?

LO QUE APRENDÍ

Elabora un mapa mental sobre la fase de implementación de un sistema de información, integra todos los aspectos relacionados en esta fase.

EXAMEN DE AUTOEVALUACIÓN

Responde verdadero (V) o falso (F).

1. La implementación es el proceso que asegura la operatividad del sistema de información y permite al usuario obtener beneficios por su operación. ( )
2. El personal técnico deberá recibir entrenamiento enfocado a las herramientas de *hardware* y *software* asociadas al sistema de información. ( )
3. Los usuarios del sistema deberán ser entrenados en los aspectos técnicos del sistema. ( )
4. Los gerentes adversos a la capitación permiten la capacitación siempre y cuando esté justificada. ( )
5. Los gerentes indiferentes justifican la falta de capacitación por los costos que implica. ( )
6. Todo nuevo sistema de información al ser implementado conlleva un cambio en los procesos de la empresa. ( )
7. La implementación del sistema siempre será acompañada de la documentación necesaria para entender el funcionamiento del sistema y para su correcta operación. ( )
8. Cuando se implementa un sistema de información nuevo, éste generalmente es independiente del *hardware* en el que se instala. ( )
9. Para establecer de manera adecuada los requisitos de *software*, es deseable generar un documento que permita analizar su entorno y funcionalidad para determinar qué *software* es el indicado para su instalación. ( )
10. El principal elemento a considerar dentro de los requisitos de *software* de un sistema es el sistema operativo con el cual trabajará. ( )

MESOGRAFÍA

BIBLIOGRAFÍA RECOMENDADA

Autor	Capítulo	Páginas
Pressman	31	559-572
Sánchez Garreta	5	69-80

BIBLIOGRAFÍA BÁSICA (REFERENCIAS)

Sánchez Garreta. *Ingeniería de proyectos informáticos: actividades y procedimientos*. Universitat Jaume I, 2003,165 pp.

Pressman, Martínez, Segovia. *Ingeniería del* software*. Un enfoque práctico* 5.ª ed. España: McGraw-Hill, 2002, 601 pp.

BIBLIOGRAFÍA COMPLEMENTARIA (TEMARIO ANALITICO)

1. Bardou, Louis. *Mantenimiento y soporte logístico de los sistemas informáticos.* México: Alfa Omega-Marcombo, 2004, 292 pp.
2. Bochinno, William A. *Sistemas de información para la administración, técnicas e instrumentos.* México: Trillas, 2002, 403 pp.
3. Bonsón, Enrique. *Tecnologías inteligentes para la gestión empresarial*. México: Alfa Omega-Rama, 2002, 258 pp.
4. Cornella, Alfons. *Información digital para la empresa, una introducción a los servicios de información electrónica.* México: Alfa Omega-Marcombo, 2004, 196 pp.
5. Lardent, Alberto R. *Sistemas de información para la gestión empresarial, procedimientos, seguridad y auditoría*. Buenos Aires: Pearson Education-Prentice Hall, 2001. 443 pp.

6. Levine, Guillermo. *Computación y programación moderna*. México: Addison Wesley, 2000, 609 pp.
7. Long, Nancy y Larry Long. *Introducción a las computadoras y a los sistemas de información*, *edición Internet.* México: Prentice Hall, 1999, 416 pp.
8. McLeod, Raymond, Jr. *Sistemas de información gerencial* 7.ª ed. México: Prentice Hall, 1999, 688 pp.
9. Oz, Effy. *Administración de sistemas de información*, 2.ª ed. México: Thomson-Learning, 2001, 578 pp.
10. Peña R., Baeza-Yates, R. y Rodríguez, J. *Gestión digital de la información. de bits a bibliotecas digitales y la web.* México: Alfa Omega-Rama, 2004, 464 pp.
11. Piattini, Mario, J. Antonio Calvo-Manzano, Joaquín Cervera y Luis Fernández. *Análisis y diseño detallado de aplicaciones informáticas de gestión.* México: Alfa Omega-Rama, 2004, 728 pp.
12. Stair, Ralph M. *Principios de sistemas de información*, 4.ª ed. México: Thomson Learning, 2002, 692 pp.
13. Walker, D. W. *Sistemas e información para la administración.* México: Alfa Omega-Marcombo, 2001, 360 pp.

SITIOS ELECTRÓNICOS

Sitio	Descripción
http://goo.gl/9hNTFN	Sistemas de información III, Implementación de sistemas. Por Luis Eduardo Mendoza. Universidad Simón Bolívar.
http://goo.gl/4nvYl2	Sistemas de información. Su implementación. Por Mauricio Lefcovich. En tu obra UNAM.
http://goo.gl/h41GN4	Ejemplo de un plan de implementación de un sistema de información. Universidad Francisco Gavidia, El Salvador.

UNIDAD 6. PRUEBAS

OBJETIVO ESPECÍFICO

Al finalizar la unidad, el alumno identificará los diferentes tipos de pruebas que son realizadas a un sistema antes de su liberación.

INTRODUCCIÓN

En cualquier disciplina, las pruebas ayudan a medir los resultados de los diversos tipos de sistemas. Hay pruebas de resistencia física de materiales, médicas, hasta las que se realizan al *software* y a los sistemas de información. En el caso de los sistemas de información, existen diversos tipos de pruebas que permiten a los desarrolladores determinar si el comportamiento de un sistema de información es adecuado, ya que se obtienen resultados en correspondencia con los objetivos o requisitos planteados desde el mismo proceso de análisis en el ciclo de vida de los sistemas. En todo caso, las pruebas a los sistemas de información ayudarán a determinar si el sistema no solamente ha sido construido de forma adecuada, sino que en su momento precisará si es aceptado o rechazado por los clientes.

A lo largo de esta unidad, revisaremos la gama de pruebas aplicables a los sistemas de información, que van desde la verificación de sus partes internas hasta la validación de su funcionamiento global.

LO QUE SÉ

Redacta una ficha donde describas qué son las pruebas de un sistema de información y en qué consisten.

TEMARIO DETALLADO (4 HORAS)

6. Pruebas
6.1. Pruebas de unidad
6.2. Pruebas de integración
6.3. Pruebas de validación
6.4. Pruebas del sistema
6.5. Paralelo

6. Pruebas

La fase de pruebas es una de las más críticas en el ciclo de vida de un sistema de información. A través de ésta, es posible determinar si el sistema construido cubre los requerimientos establecidos desde su concepción, ya que su cumplimiento asegurará la calidad del sistema. Cuando se está en la fase de pruebas, los ingenieros de *software* diseñan un conjunto de tareas cuyo objetivo es hacer que falle de forma intencional el sistema construido.

Objetivos de las pruebas[59]

- La prueba es el proceso de ejecución de un programa con la intención de descubrir un error.
- Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.
- Una prueba tiene éxito si descubre un error no detectado hasta entonces.

Principios de las pruebas

Antes de comenzar a diseñar las pruebas a las que será sometido un sistema de información, se tomará en cuenta lo siguiente:

- Todas las pruebas deben poder ser rastreadas hasta en uno o varios de los requisitos establecidos por los clientes. Un sistema de información de calidad cumplirá a cabalidad los requisitos establecidos por los clientes.

[59] Pressman, R. *Ingeniería del* software. *Un enfoque práctico*, 5.ª ed., p. 305.

- Todas las pruebas a ser realizadas deben ser planificadas con anticipación. La planificación de las pruebas a ser realizadas puede comenzar desde la misma fase de análisis del sistema.

- El 80 por ciento de los errores descubiertos en la fase de pruebas puede ser rastreado hasta en 20 por ciento de los módulos que integran el sistema (principio de Pareto).

- Todas las pruebas serán diseñadas para comenzar con los componentes menos significativos del sistema, e incrementarán su nivel hasta llegar a los más significativos. En muchas ocasiones, los errores más graves están en los detalles menos significativos de los sistemas.

- En un sistema de información, es imposible realizar pruebas de manera exhaustiva, lo anterior debido a la complejidad que conlleva una revisión módulo a módulo en cada uno de sus componentes. Por tato, es deseable hacer pruebas que midan la funcionalidad completa de los módulos, con lo que se cubre la mayor parte de los componentes.

- Las pruebas serán realizadas en equipos independientes a donde fue programado el sistema, con el objetivo de encontrar, en condiciones lo más cercanas a las de operación en su ámbito real, la mayor cantidad de errores posibles para su corrección.

Facilidad de prueba

Dentro de un diseño ideal de un sistema de información, los ingenieros de *software* deben concebir el sistema teniendo en mente la fase de pruebas, lo que permitirá a los *tésters* realizar las pruebas de una forma más eficiente. La realidad muchas veces nos

indica lo contrario, por ello, James Bach[60] describe la facilidad de prueba de la siguiente manera: “La facilidad de prueba del *software* es simplemente la facilidad con la que se puede probar un programa de computadora. Como la prueba es tan profundamente difícil, merece la pena saber qué se puede hacer para hacerlo más sencillo. A veces los programadores están dispuestos a hacer cosas que faciliten el proceso de prueba y una lista de comprobación de posibles puntos de diseño, características, etcétera, puede ser útil a la hora de negociar con ellos”.

Cuando se diseñan casos de pruebas, se debe contar con elementos tangibles que permitan medir su éxito. En este caso, hay una serie de características para determinar qué tan fácil es un sistema para ser probado[61]:

1. Operatividad. “Cuanto mejor funcione, más eficientemente se puede probar”.

- El sistema tiene pocos errores (los errores añaden sobrecarga de análisis y generación de informes al proceso de prueba).
- Ningún error bloquea la ejecución de las pruebas.
- El producto evoluciona en fases funcionales (permite simultanear el desarrollo y las pruebas).

2. Observabilidad. “Lo que ves es lo que pruebas”.

- Se genera una salida distinta para cada entrada.
- Los estados y variables del sistema están visibles o se pueden consultar durante la ejecución.
- Los estados y variables anteriores están visibles o se pueden consultar (por ejemplo, registros de transacción).
- Todos los factores que afectan a los resultados están visibles. Un resultado incorrecto se identifica con facilidad.

[60] Pressman, R. *Ingeniería del* software. *Un enfoque práctico*, 5.ª ed., p. 283.
[61] Pressman, R. *Ingeniería del* software. *Un enfoque práctico*, 5.ª ed., p. 283.

- Los errores internos se detectan automáticamente a través de mecanismos de autocomprobación.
- Se informa automáticamente de los errores internos.
- El código fuente es accesible.

3. *Controlabilidad. "Cuanto mejor podamos controlar el* software*, más se puede automatizar y optimizar".*
   - Todos los resultados posibles se pueden generar a través de alguna combinación de entrada.
   - Todo el código es ejecutable a través de alguna combinación de entrada.
   - El ingeniero de pruebas puede controlar directamente los estados y las variables del *hardware* y del *software*.
   - Los formatos de las entradas y los resultados son consistentes y estructurados.
   - Las pruebas pueden especificarse, automatizarse y reproducirse convenientemente.

4. *Capacidad de descomposición. "Controlando el ámbito de las pruebas, podemos aislar más rápidamente los problemas y llevar a cabo mejores pruebas de regresión".*
   - El sistema *software* está construido con módulos independientes.
   - Los módulos del *software* se pueden probar independientemente.

5. *Simplicidad. "Cuanto menos haya que probar, más rápidamente podremos probarlo".*
   - Simplicidad funcional (por ejemplo, el conjunto de características es el mínimo necesario para cumplir los requisitos).

- Simplicidad estructural (por ejemplo, la arquitectura es modularizada para limitar la propagación de fallos).
- Simplicidad del código (por ejemplo, se adopta un estándar de código para facilitar la inspección y mantenimiento).

6. *Estabilidad. "Cuanto menos cambios, menos interrupciones a las pruebas".*
   - Los cambios del *software* son infrecuentes.
   - Los cambios del *software* están controlados.
   - Los cambios del *software* no invalidan las pruebas existentes.
   - El *software* se recupera bien de los fallos.

7. *Facilidad de comprensión. "Cuanta más información tengamos, más inteligentes serán las pruebas".*
   - El diseño se ha entendido perfectamente.
   - Las dependencias entre los componentes internos, externos y compartidos se han entendido perfectamente.
   - Se han comunicado los cambios del diseño.
   - La documentación técnica es accesible de manera instantánea.
   - La documentación técnica está bien organizada.
   - La documentación técnica es específica y detallada.
   - La documentación técnica es exacta.

Los atributos antes mencionados pueden ser documentados para ser empleados en la fase de configuración del *software*.

6.1. Pruebas de unidad

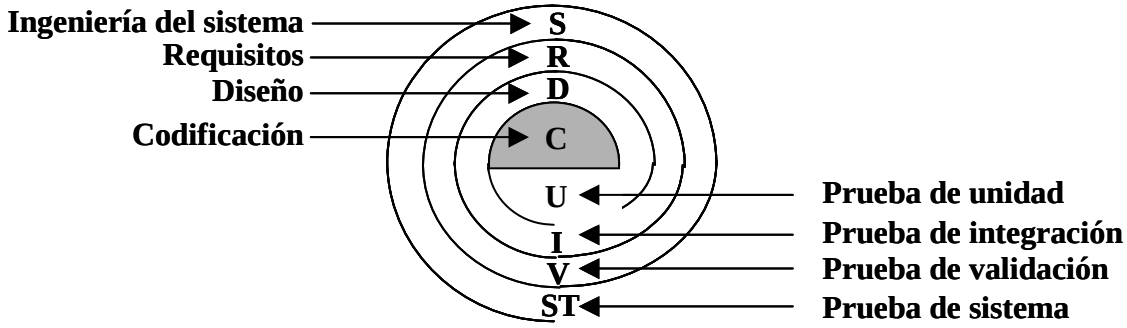


Figura 6.1. Pruebas de unidad.[62]

Las pruebas unitarias o de unidad tienen como objetivo verificar la funcionalidad y estructura de cada componente individualmente una vez que ha sido codificado[63]. Estas pruebas comienzan realizando una verificación de los componentes más pequeños del sistema hasta llegar a la totalidad del *software*. En otras palabras, llevan a cabo una verificación de los subprogramas, subrutinas, funciones, procedimientos, etcétera, hasta realizar una verificación completa del funcionamiento del sistema en su totalidad.

Razones principales para realizar las pruebas de unidad:

- Manejan los elementos de forma combinada, ya que centran su atención en las partes más pequeñas del sistema.

- Ayudan a eliminar errores fácilmente, pues al comenzar con las partes más básicas del sistema, al ser detectados los errores, éstos son asociados inmediatamente a un elemento en particular, lo que agiliza su corrección.

[62] Pruebas de unidad. Disponible en http://goo.gl/t4zx4T. Recuperado: 16/10/13.
[63] Pruebas de unidad. Disponible en http://goo.gl/t4zx4T. Recuperado: 16/10/13.

- Permiten realizar pruebas en diferentes módulos del sistema de forma simultánea, lo que acelera el proceso de detección y corrección de errores.

Los elementos para llevar a cabo las pruebas de unidad son las especificaciones para el módulo a ser probado y su código fuente. En el primer caso, las especificaciones son necesarias, pues en ellas se definen los parámetros de entrada y salida del módulo y su funcionamiento. Y el código fuente servirá para poder detectar el lugar exacto donde suceda el error.

Procedimiento para la realización de las pruebas de unidad:

- Analizar la lógica del módulo usando uno o más de los métodos de caja blanca.
- Completar los casos de prueba aplicando métodos de caja negra a la especificación del módulo.[64]

6.2. Pruebas de integración

Las pruebas de integración se orientan a la verificación de la forma como los diferentes módulos que integran el sistema interactúan entre sí, una vez que las pruebas de unidad han comprobado el funcionamiento individual de cada módulo. Es decir, corroboran el funcionamiento correcto de la interfaces internas y externas de los módulos, para comprobar que el sistema se ajusta a lo requerido por los clientes.

[64] Pressman, R. *Ingeniería del* software. *Un enfoque práctico*, 5.ª ed., p. 311.

Tipos fundamentales de integración[65]:

- *Integración incremental*. Se combina el siguiente componente que se debe probar con el conjunto de componentes que ya están probados, y se va incrementando progresivamente el número de componentes a probar.

- *Integración no incremental.* Se prueba cada componente por separado y luego se incorporan todos a la vez realizando las pruebas pertinentes.

6.3. Pruebas de aceptación (validación)

Las pruebas de aceptación tienen el objetivo de validar el sistema, es decir, revisan que el sistema en su totalidad cumpla con los requisitos de funcionamiento establecidos por el cliente desde el inicio. Esto, al final de cuentas, determinará si el sistema es aceptado o no por el cliente desde el punto de vista de funcionalidad y rendimiento. Por lo general, estas pruebas son definidas por los usuarios finales, aunque su implementación es responsabilidad de los desarrolladores del sistema; mientras que la ejecución de pruebas y aprobación final son realizadas por el usuario del sistema.

El resultado de las pruebas de validación es medido desde el punto de vista de las entradas y salidas del sistema. Es decir, se confirmará que los datos ingresados al sistema sean los solicitados por el cliente; y los resultados entregados sean, de igual forma, los que el cliente espera recibir. Esto llevará a la validación de los requisitos funcionales y no funcionales establecidos desde el inicio del desarrollo.

[65] Metodología MÉTRICA versión 3. Técnicas y Prácticas. Ministerio de Administraciones Públicas, 2002. Disponible en http://goo.gl/T9m1N1.

6.4. Pruebas del sistema

La intención de las pruebas del sistema es identificar incongruencias entre los objetivos del sistema y su funcionamiento. Se enfocan principalmente a los errores que pudieron ser cometidos al plasmar los requerimientos en el diseño mismo del sistema; en otras palabras, verifican la cantidad de errores a lo largo del proceso de desarrollo y miden su impacto en el sistema. No se centran en las funciones o módulos del sistema (para esto son realizadas la pruebas de unidad e integridad), más bien ponen énfasis en confirmar si el sistema en general cumple con los objetivos o requerimientos establecidos desde el comienzo.

Aplicaciones principales[66]:

Las pruebas de sistema no se limitan a los sistemas. Si el producto es un programa, la prueba del sistema es el proceso de procurar demostrar cómo el programa, en su totalidad; no resuelve sus objetivos o requerimientos.

- Las pruebas de sistema no son posibles de realizar si los requerimientos no se encuentran establecidos por escrito y no son cuantificables.

- Las pruebas de sistema son pruebas de integración global del sistema de información. A través de ellas, se comprueba al sistema en su totalidad y la forma de interacción entre otros sistemas que estén asociados con él, lo que permite medir su funcionamiento en un entorno muy similar al que tendrá una vez puesto en marcha.

[66] Herrera González, C. *Estrategias de aplicación de pruebas de unidad, integración, sistemas y aceptación.* Instituto Tecnológico Superior de Libres, Puebla. Disponible en http://goo.gl/eLGvyd. Recuperado: 16/10/2013.

Tipos de pruebas

- *De comunicaciones.* Comprueban la interacción entre los módulos y el funcionamiento de las diversas interfaces que los componen, ya sea de forma remota o local. Adicionalmente, se verifica el funcionamiento de las interfaces con los usuarios.

- *Pruebas de rendimiento.* Tienen como fin comprobar que los tiempos de respuesta del sistema están en un intervalo aceptable por los usuarios.

- *Pruebas de recuperación.* Hacen fallar al sistema, con lo que es posible medir su capacidad para recuperarse de dichas fallas y si su funcionamiento continúa siendo el adecuado.

- *Pruebas de volumen.* Consisten en incrementar el volumen de información que el sistema recibe, acercándolo lo más posible al volumen de información esperado cuando esté en funcionamiento, con lo que es posible comprobar su capacidad de respuesta a esa cantidad de trabajo.

- *Pruebas de sobrecarga.* Saturan al sistema con grandes volúmenes de información, llevándolo al límite de sus capacidades y recursos, con el objetivo de establecer los límites dentro de los cuales el sistema deja de cumplir con los requisitos establecidos.

- *Pruebas de tensión.* Sobrecargan al sistema con grandes volúmenes de información en periodos cortos, a fin de ubicar una tolerancia del sistema en relación con el tiempo. Un ejemplo de este tipo de pruebas es cuando se le pide a una persona leer una cierta cantidad mínima de palabras dentro de un minuto:

si la persona cumple con lo solicitado, es aprobado; de lo contrario, sería rechazado.

- *Pruebas de disponibilidad de datos.* Consisten en verificar si el sistema es capaz de mantener la integridad de los datos cuando se presenten fallas.

- *Pruebas de facilidad de uso.* Buscan determinar si los usuarios se sienten cómodos al realizar su trabajo con el sistema, verificando si éste cumple con sus necesidades; si las interfaces son de fácil manejo, si los datos pueden ser ingresados de forma sencilla y si los resultados son los deseados.

- *Pruebas de operación.* Consisten en verificar si el funcionamiento del sistema es adecuado comprobando si no hay problemas al iniciarlo o reiniciarlo, si los módulos trabajan de forma adecuada, etcétera.

- *Pruebas de entorno.* Su objetivo es verificar que el sistema interactúa de forma correcta con otros sistemas de información asociados a éste.

- *Pruebas de seguridad.* Se orientan a verificar que solamente las personas autorizadas tengan acceso al sistema de información, validando los mecanismos de control de acceso.

- *Pruebas de usabilidad.* Se enfocan a la forma en que los usuarios trabajan con el sistema, verificando si se sienten cómodos al operarlo, si cumple sus expectativas, etcétera. El problema con este tipo de pruebas es que, al ser enfocadas al factor humano, tienden a ser subjetivas.

- *Pruebas de almacenamiento.* Se enfocan a medir la capacidad del sistema para almacenar datos, ya sea de forma temporal o permanente en las memorias

principal y secundaria del equipo en el que se encuentra instalado, para verificar el cumplimiento de los requerimientos establecidos.

- *Pruebas de configuración.* Se enfocan a validar la compatibilidad del sistema con los componentes de *hardware* y *software* con los interactuará, por ejemplo, el sistema operativo, el manejador de base de datos, las tarjetas de red, etcétera.

- *Pruebas de instalación.* Miden la capacidad de un sistema para ser instalado correctamente en distintas plataformas, permitiendo a los diseñadores generar programas de instalación automáticos para las plataformas donde se desea instalar el sistema. La mayoría de los diseñadores de *software* suelen generar programas instaladores de acuerdo con el tipo de sistema operativo o *hardware* instalado en el equipo.

- *Pruebas de documentación.* Revisan que todos los procesos realizados durante el desarrollo del sistema estén bien documentados y contengan a detalle la información de cada fase del sistema, sus requerimientos, resultados de pruebas, etcétera. La documentación es empleada a menudo para diseñar las pruebas a las que será sometido el sistema; una documentación correcta permite a las empresas tener independencia entre el sistema y sus desarrolladores.

6.5. Paralelo

Estas pruebas pueden ser llevadas a cabo de manera simultánea en diversos sistemas. En el caso de algunos sistemas de misión crítica, es necesario contar con sistemas de respaldo o espejo, que permiten, si falla el sistema principal, que se mantenga el servicio prestado por el sistema. Si las salidas obtenidas por diversos sistemas son idénticas, se asumen correctas; de lo contrario, se analizan todas las aplicaciones para determinar el problema. Se debe tener cuidado con este tipo de prueba, pues si el error se encuentra en la fase de análisis, éste permanecerá en todos los sistemas espejo, y si

las pruebas dan resultados similares para todos los sistemas, pero erróneos, la prueba no podrá detectar el error.

RESUMEN DE LA UNIDAD

Las fases de pruebas conforman una parte fundamental en la construcción de un sistema de información, ya que permiten a los desarrolladores o ingenieros de *software* detectar y, de ser posible, corregir problemas en un sistema de información al ser expuesto a condiciones de funcionamiento cercanas a las reales. Existe una gama diferente de pruebas aplicables a un sistema de información, todas con los mismos objetivos, detectar problemas en diferentes ámbitos del sistema, desde su funcionamiento interno hasta su forma de interactuar con su entorno operativo. Un buen diseño de sistemas de información ha de considerar las posibles pruebas que le serán aplicadas desde su concepción misma.

Pruebas más importantes aplicables a los sistemas de información:

- *Pruebas de unidad.* Enfocadas a verificar que el sistema funcione de forma correcta desde su parte más insignificante hasta su conjunto; se concentran en el funcionamiento interno del sistema de información.

- *Pruebas de integración.* Encauzadas principalmente a validar la forma en que los componentes o módulos internos del sistema interactúan entre sí.

- *Pruebas de validación.* Validan que los objetivos del sistema o sus requisitos funcionales y no funcionales sean cubiertos en su totalidad.

- *Pruebas de sistema.* Verifican o validan que el sistema en general cumpla con los requerimientos establecidos, realizando pruebas tanto a su diseño como a su funcionalidad e interacción con su medio ambiente.

GLOSARIO DE LA UNIDAD

Integridad de datos

Se refiere a que un dato debe permanecer sin alteraciones al ser almacenado en una base de datos o cualquier medio de almacenamiento para su uso posterior.

Misión crítica

Procesos o tareas que son el soporte principal para las operaciones cotidianas de una organización.

Pruebas de caja negra y caja blanca

En programación, se denomina “cajas blancas” a un tipo de pruebas de *software* que se realiza sobre las funciones internas de un módulo. Y las “cajas negras” ejercitan los requisitos funcionales desde el exterior del módulo.

Requisito o requerimiento

Condición o facultad que necesita un usuario para resolver un problema; o condición o facultad que debe poseer un sistema o un componente de un sistema para satisfacer una especificación, estándar, condición de contrato u otra formalidad impuesta documentalmente. También se refiere al documento que recoge lo anterior.

Validación

Confirmación, mediante examen y aportación de pruebas objetivas, de que se cumplen los requisitos concretos para un uso determinado. Responde a la pregunta *¿estamos construyendo el producto correcto?*

Verificación

Confirmación, mediante examen y aportación de pruebas objetivas, de que se cumplen los requisitos específicos. Responde a la pregunta: *¿estamos construyendo correctamente el producto?*

ACTIVIDADES DE APRENDIZAJE

ACTIVIDAD 1

Realiza una investigación en Internet sobre tres ejemplos de pruebas de unidad aplicables a un sistema de información. Con base en lo anterior, escribe a manera de resumen en qué consisten las pruebas que investigaste y en qué casos son aplicables.

ACTIVIDAD 2

Discute con tus compañeros sobre la importancia de las pruebas de unidad, con base en las siguientes preguntas:

1. ¿Cuál es objetivo de las pruebas de unidad?
2. ¿Cuáles son sus ventajas?
3. ¿Por qué es importante considerarlas dentro de un plan de pruebas de un sistema de información?

ACTIVIDAD 3

Realiza una investigación sobre las diferencias y similitudes de las pruebas de caja negra y las de integridad y validación. Escribe una síntesis de tu investigación en dos cuartillas y menciona un ejemplo de cada tipo de prueba.

ACTIVIDAD 4

Elabora un cuadro sinóptico sobre las diferentes pruebas de sistema que pueden ser realizadas a un sistema de investigación y plantea un ejemplo de cada una.

CUESTIONARIO DE REFORZAMIENTO

Responde las siguientes preguntas.

1. ¿Qué es una prueba?
2. ¿En qué consiste la fase de pruebas en el ciclo de vida de los sistemas de información?
3. ¿Qué son las pruebas de unidad?
4. ¿Qué son las pruebas de integridad?
5. ¿Cómo se validan los requisitos a través de las pruebas de un sistema de información?
6. ¿Qué significa que un sistema debe tener facilidad de prueba?
8. Escribe tres ejemplos de pruebas de sistema.
9. ¿Qué es una prueba de validación?
10. ¿Cuál es el objetivo de las pruebas de sistema?

LO QUE APRENDÍ

Elabora un mapa mental donde plantees las diversas pruebas que pueden ser aplicadas a los sistemas de información y sus características principales.

EXAMEN DE AUTOEVALUACIÓN

Responde verdadero (V) o falso (F).

1. A través de la fase de pruebas, es posible determinar si el sistema construido cumple con los requerimientos establecidos por sus desarrolladores. ( )
2. Todas las pruebas deben poder ser rastreadas hasta uno o varios de los requisitos establecidos por los clientes. ( )
3. El 80 por ciento de los errores descubiertos en la fase de pruebas puede ser rastreado hasta en un 20 por ciento de los módulos que integran el sistema. ( )
4. Las pruebas unitarias tienen como objetivo verificar la funcionalidad y estructura general del sistema una vez que ha sido codificado. ( )
5. Las pruebas de integración se orientan a la verificación de la forma como los diferentes módulos trabajan de manera individual. ( )
6. En las pruebas de integración no incremental, se prueba cada componente por separado y luego se integran todos a la vez realizando las pruebas pertinentes. ( )
7. Las pruebas de aceptación son definidas por los usuarios finales, aunque su implementación es responsabilidad de los desarrolladores del sistema. ( )
8. Las pruebas del sistema tienen como objetivo identificar incongruencias entre los objetivos del sistema y su funcionamiento. ( )
9. Las pruebas de rendimiento tienen como objetivo comprobar que los datos de salida del sistema cuentan con un formato aceptable por los usuarios. ( )
10. Las pruebas de sobrecarga consisten principalmente en saturar al sistema con grandes volúmenes de información, llevándolo al límite de sus capacidades. ( )

MESOGRAFÍA

BIBLIOGRAFÍA RECOMENDADA

Autor	Capítulo	Páginas
Pressman	8	131-148
Peña	5	74

BIBLIOGRAFÍA BÁSICA (REFERENCIAS)

Pressman Martínez, Segovia. *Ingeniería del* software*. Un enfoque práctico*, 5.ª ed. España: McGraw-Hill, 2002, 601 pp.

Peña. *Ingeniería de software: una guía para crear sistemas de información*. México: IPN, 2006, 120 pp.

BIBLIOGRAFÍA COMPLEMENTARIA (TEMARIO ANALÍTICO)

1. Bardou, Louis. *Mantenimiento y soporte logístico de los sistemas informáticos.* México: Alfa Omega-Marcombo, 2004, 292 pp.
2. Bochinno, William A. *Sistemas de información para la administración, técnicas e instrumentos.* México: Trillas, 2002.
3. Bonsón, Enrique. *Tecnologías inteligentes para la gestión empresarial*. México: Alfa Omega-Rama, 2002, 258 pp.
4. Cornella, Alfons. *Información digital para la empresa, una introducción a los servicios de información electrónica.* México: Alfa Omega-Marcombo, 2004, 196 pp.
5. Lardent, Alberto R. *Sistemas de información para la gestión empresarial, procedimientos, seguridad y auditoría*. Buenos Aires: Pearson Education-Prentice Hall, 2001.

6. Levine, Guillermo. *Computación y programación moderna*. México: Addison Wesley, 2000.
7. Long, Nancy y Larry Long. *Introducción a las computadoras y a los sistemas de información*, *edición Internet.* México: Prentice Hall, 1999, 416 pp.
8. McLeod, Raymond, Jr. *Sistemas de información gerencial,* 7.ª ed. México: Prentice Hall, 1999, 688 pp.
9. Oz, Effy. *Administración de sistemas de información*, 2.ª ed. México: Thomson-Learning, 2001, 578 pp.
10. Peña R., Baeza-Yates, R. y Rodríguez, J. *Gestión digital de la información. de bits a bibliotecas digitales y la web.* México: Alfa Omega-Rama, 2004, 464 pp.
11. Piattini, Mario, J. Antonio Calvo-Manzano, Joaquín Cervera y Luis Fernández. *Análisis y diseño detallado de aplicaciones informáticas de gestión.* México: Alfa Omega-Rama, 2004, 728 pp.
12. Stair, Ralph M. *Principios de sistemas de información*, 4.ª ed. México: Thomson Learning, 2002, 692 pp.
13. Walker, D. W. *Sistemas e información para la administración.* México: Alfa Omega-Marcombo, 2001, 360 pp.

SITIOS ELECTRÓNICOS

Dirección electrónica	Descripción
http://goo.gl/13ADWR	Documento sobre pruebas a sistemas de información de la Universidad de Castilla-La Mancha, España, por Macario Polo Usaola.
http://goo.gl/kjNPjC	Presentación electrónica sobre pruebas a sistemas de información. Curso de ingeniería de *software*. Departamento de lenguajes y sistemas informáticos. Universidad de Granada. España.

UNIDAD 7
MANUAL DE USUARIO

OBJETIVO ESPECÍFICO

Al terminar la unidad, el alumno reconocerá las características principales de los manuales primordiales que deben elaborarse junto a un sistema de información.

INTRODUCCIÓN

Una vez terminado el desarrollo e implementación de un sistema de información, es necesario desarrollar una serie de manuales del sistema que deberán ser entregados en conjunto a los clientes: manuales técnicos, de usuario y de operación. El manual técnico considera de manera detallada la documentación generada a lo largo de todo el proceso de desarrollo e implementación del sistema, desde la documentación de los requisitos establecidos en la fase de análisis, hasta los resultados obtenidos en las pruebas e implementación del sistema. El manual de usuario contiene las instrucciones necesarias para que el usuario final pueda utilizar de forma adecuada el sistema. Y el manual de operación comprende la información necesaria para emplear las funciones que integran al sistema de forma correcta.

En esta unidad, revisaremos a detalle cada uno de estos documentos y su empleo en la planificación de las actividades de mantenimiento del sistema.

LO QUE SÉ

Elabora una ficha en un procesador de texto, donde describas las características que debe tener un manual de usuario de un sistema de información según tu criterio.

TEMARIO DETALLADO (8 HORAS)

7. Manual de usuario
7.1. Documentación técnica
7.2. Manual de usuario del sistema
7.3. Manual de operación del sistema
7.4. Planeación del mantenimiento del sistema
7.5. Control del mantenimiento del sistema

7. Manual de usuario

Cuando es desarrollado un sistema de información, debe ser complementado con documentación necesaria que exponga de forma detallada sus características, funciones, estructura, entre otros elementos, además de explicar a los usuarios cómo operarlo. La documentación es un proceso realizado a lo largo de todo el ciclo de vida del sistema, una tarea tan necesaria como la construcción del sistema mismo. De cierta manera, mientras la estructura de programación o código fuente muestra la forma como el sistema se estructura y opera, la documentación explica el porqué de sus funciones.

En el caso de programas de cómputo sencillos, generalmente suele bastar con el código fuente para entender su funcionamiento. En cambio, tratándose de sistemas de información mayores, éstos pueden ser tan complejos que sería prácticamente imposible comprender su funcionamiento sólo a través de un código fuente.

Los sistemas de información deben ser concebidos con la idea de poder ser modificados en un futuro no muy lejano. Y en muchas ocasiones, los desarrolladores originales de un sistema no siempre están disponibles para realizar dichas modificaciones; por eso es fundamental la documentación generada a lo largo del desarrollo del sistema, pues ella permitirá que un nuevo equipo de desarrolladores retome el sistema y haga las modificaciones pertinentes.

Por lo general, la documentación se divide en dos: un manual técnico, con toda la información necesaria para un desarrollador; y un manual de usuario, con las instrucciones para los usuarios finales que les ayudan a operar el sistema de forma adecuada.

7.1. Documentación técnica

El manual técnico es el conjunto de documentos de carácter técnico recabados a lo largo del proceso de desarrollo del sistema, que describe con profundidad sus características técnicas como la documentación de los requisitos, diagramas para el análisis y diseño del sistema, código fuente, etcétera; así como su funcionamiento completo a detalle. Este documento, por su naturaleza, está destinado a ser empleado por ingenieros de *software*, ya que éste les permite comprenderlo en su totalidad y realizarle alguna modificación, o darle mantenimiento.

Elementos que integran un manual técnico[67]:

a. Propósitos y funcionamiento del programa
b. Esquema lógico del funcionamiento del programa
c. Nuevos tipos definidos, principales variables globales, archivos utilizados, etcétera
d. Explicación de fórmulas y procesos complejos
e. Especificación de datos de entrada y datos de salida.
f. Formato de los archivos y las pantallas
g. Datos utilizados en las pruebas
h. Puntos débiles y expansiones futuras
i. Guía de referencia de rutinas
j. Diagramas
k. Listado del programa fuente
l. Otras características técnicas

El manual técnico suele quedar a resguardo del encargado del área de sistemas de una empresa u organización, con el objetivo de tenerlo a disposición cuando se requiera

[67] Ozuna, M. *Documentación técnica de un programa para Arqhys Arquitectura*. Disponible en http://goo.gl/VAKPbs. Recuperado: 2/04/2013.

hacer alguna modificación o verificar su instalación. Por otro lado, parte de la información importante que contiene el manual técnico son la descripción y tamaño que deben tener los archivos ejecutables, ya que un cambio en su tamaño podría sugerir que ha sido modificado ya sea por un virus o por alguna persona que tuvo acceso al código fuente.

Otro aspecto de utilidad es el referente a los resultados de las pruebas realizadas al sistema, que por lo regular documentan los puntos débiles del sistema. Estos puntos débiles pueden ayudar a los desarrolladores a identificar las posibles fallas críticas y anticiparlas mediante una actualización del sistema una vez puesto en marcha.

La siguiente figura (7.1) muestra una plantilla que cuenta con los elementos mínimos y algunos ejemplos para elaborar el manual técnico.

Plantilla general para un manual técnico

☞ *Este es un modelo de los temas que puede incluir en el manual del usuario, algunas de las secciones incluyen una redacción propuesta para abordar ese punto en particular, y otras sólo mencionan qué deberá incluirse.*

Indice

☞ *Este es el índice general, pueden agregársele otros capítulos.*

1. Propósitos y funcionamiento del programa

Los próposito del programa son:

Figura 7.1. Ejemplo de contenido de un manual técnico.[68]

7.2. Manual de usuario del sistema

El manual de usuario es un documento dirigido a los usuarios finales del sistema, en el que se indican los pasos necesarios para que los usuarios puedan emplear y operar de forma adecuada el sistema de información. Asimismo, contiene los objetivos del

[68] Vasconcelos. *Manual para programar*, p. 112.

sistema, así como los diferentes menús u opciones que lo conforman y sus características principales. Debe ser redactado de la manera más simple, evitando terminología técnica (en caso de ser empleada, se explicará e forma sencilla), pues los usuarios del sistema no son necesariamente expertos en el área informática.

Principales elementos que conforman el manual de usuario[69]:

- Descripción clara y amplia del objetivo del programa
- Requerimientos mínimos de *hardware*
- Sistema operativo
- Pasos necesarios para instalarlo por primera vez y para desinstalarlo.
- Pasos necesarios para ejecutarlo
- Descripción de opciones, comandos y menús
- Descripción y muestra de salidas que se pueden obtener
- Ejemplos de operación
- Posibles errores y cómo solucionarlos
- Nombres de los archivos que se incluyen en las unidades de almacenamiento y breve descripción de los mismos
- Glosario

A pesar de ser un documento relativamente simple, el manual de usuario es uno de los documentos que más trabajo cuesta escribir a un ingeniero de *software*, debido a que, al estar tan familiarizado con el sistema de información desde su concepción misma, muchas veces encuentra complicado aterrizar todos esos conceptos en un manual escrito de forma simple, o suele olvidar al usuario final.

[69]Basado en Cohuo Cuevas, J. A. Unidad 3. Requisitos de los elementos de los manuales administrativos. Disponible en http://goo.gl/mSfPOI. Recuperado: 5/5/2013.

7.3. Manual de operación del sistema

El manual de operación es un documento que describe a detalle la forma de emplear cada una de las partes que integran el sistema, con base en los requisitos funcionales establecidos para su diseño. Por lo general, este tipo de manual está dirigido a los usuarios directos del sistema, quienes utilizarán la mayor parte de sus funciones.

El manual de operación describirá los siguientes aspectos:

- Características generales de aplicación.
- Quiénes son los usuarios y las relaciones que tiene la aplicación con los procesos de negocio (procesos del dominio de la aplicación) que los usuarios ejecutan.
- La interfaz usuario/sistema de la aplicación.
- Cada una de las funciones de la aplicación: cómo activarla, qué datos debe proporcionar el usuario, qué datos o información produce el sistema, procesos de negocio apoya, y qué mensajes de advertencia o error produce la aplicación.

7.4. Planeación del mantenimiento del sistema

Uno de los objetivos principales de cualquier sistema de información de calidad debe ser asegurar la disponibilidad de la información almacenada y generada por dicho sistema. Para esto es necesario realizar un plan de mantenimiento adecuado sobre sus componentes principales y de este modo asegurar su funcionamiento óptimo y garantizarlo a largo plazo.

Planificar las acciones de mantenimiento de un sistema ayuda a las empresas a evitar detener sus operaciones a causa de un fallo, que a su vez conllevaría costos asociados tanto a la productividad como a la reparación del mismo sistema. Además, a partir de un

plan de mantenimiento, las empresas pueden realizar las tareas de mantenimiento en los tiempos señalados sin afectar la productividad laboral.

Dentro de las acciones de mantenimiento, se tomará en cuenta el entorno externo de la empresa: cambios en las legislaciones, políticas laborales, etcétera. En este orden, algunos de los requisitos externos a considerar se listan a continuación[70]:

- *Recomendaciones del fabricante*. El fabricante de sus objetos técnicos puede recomendar ciertos procedimientos para garantizar que los objetos funcionen siempre de manera óptima.
- *Requisitos legales.* Pueden existir leyes de protección del trabajo o leyes sobre la seguridad de objetos que le obliguen a realizar el mantenimiento preventivo de su sistema técnico periódicamente.
- *Requisitos del entorno.* Un mantenimiento planificado eficaz puede ayudar a evitar paradas que podrían conllevar peligro para el entorno.

Conservar la calidad de los sistemas de información en una empresa debe ser un motivo suficientemente fuerte para planificar las tareas de mantenimiento. Tener en perfecto funcionamiento los puntos medulares de un sistema de información y verificar de forma periódica los resultados que entrega, son factores que ayudan a sostener su calidad. Para este propósito, el manual técnico es de vital importancia, ya que dentro de éste se describen el funcionamiento de cada componente y los posibles puntos débiles que afecten la calidad del sistema.

[70] Biblioteca SAP. Planificación de mantenimiento (CS-AG/PM-PRM-MP) Disponible en http://goo.gl/fK7grq. Recuperado: 14/10/2013.

7.5. Control del mantenimiento del sistema

Los planes de mantenimiento son especialmente útiles para poder establecer fechas de actividades de mantenimiento y los alcances que tendrá cada tarea planificada. Los resultados y actividades deberán ser documentados para utilizarlos en futuras planificaciones y mantener un histórico de las actividades realizadas y los módulos donde fueron efectuadas.

Asimismo, los planes de mantenimiento aseguran que los objetivos y tiempos establecidos para cada actividad sean alcanzados de forma adecuada, lo que garantiza la funcionalidad óptima del sistema y permite un control sobre los cambios. En esta línea, para realizar de forma adecuada las tareas de mantenimiento planificadas y llevar un control sobre las mismas, es fundamental seguir estas recomendaciones[71]:

- Para la planificación global de medidas
  - Aviso de mantenimiento
  - Aviso de servicio
- Para la planificación detallada de medidas
  - Orden de mantenimiento
  - Orden de servicio
- Para la planificación detallada de medidas y el historial de la avería procesada en el aviso
  - Aviso de mantenimiento y orden de mantenimiento simultáneamente
  - Aviso de servicios y orden de servicio simultáneamente
- Para el servicio al cliente
  - Planes de mantenimiento con referencia a un contrato marco
- Para la gestión de calidad mediante el enlace a características de inspección (administración de calidad)

[71] Biblioteca SAP. Planificación de mantenimiento. Disponible en http://goo.gl/a5xPFj. Recuperado: 10/07/2013.

  - Lotes de inspección
- Para obtención de servicio en la compra:
  - Hojas de entrada de servicios
- Resumen gráfico de la programación de mantenimiento
- Lista con fechas de mantenimiento calculadas
- Visualización de costes para planes de mantenimiento preventivo
- Archivar los planes de mantenimiento preventivo

Mantenimiento a varios niveles

Es posible realizar las tareas de mantenimiento a manera de capas o niveles, es decir, indicando los puntos de forma jerárquica que serán afectados por cada actividad, lo que posibilita establecer de manera más eficiente los objetivos de las tareas de mantenimiento. Aquí, el manual técnico es de gran ayuda para identificar los componentes que serán afectados en cada tarea de mantenimiento, y para establecer sus niveles:

- Equipo
- Ubicaciones técnicas
- Materiales
- Materiales y números de serie
- Conjuntos

Además, la planificación por niveles permite dar mantenimiento de forma más efectiva al sistema o aplicación y a los módulos o subrutinas. Al separar las actividades de forma específica, es fácil aplicar actualizaciones o correcciones al sistema de forma invisible para los usuarios sin afectar el funcionamiento principal del sistema.

RESUMEN DE LA UNIDAD

La documentación de un sistema de información es parte integral de las actividades que se desarrollan a lo largo del ciclo de vida de un sistema de información. Cada proceso y actividad debe de ser documentado de forma adecuada, lo que dará como resultado final un manual técnico que compile toda la documentación generada asociada con el sistema.

El manual técnico es un documento con toda la documentación de carácter técnico derivada del desarrollo del sistema: diagramas de flujo de datos, requerimientos, casos de uso, diccionarios de datos, etcétera. Ayuda a garantizar la continuidad del funcionamiento del sistema y la integridad de la información que procesa y almacena; es decir, su contenido hará que no se dependa de los desarrolladores originales del sistema. Este manual es una herramienta de gran utilidad en la planificación y control de las actividades de mantenimiento a realizar en el sistema, ya que los objetivos de dichas actividades son la mejora de los componentes del sistema de información o la corrección de errores que surjan durante su funcionamiento.

Otro manual derivado del proceso de desarrollo del sistema es el del usuario, donde se deben describir los pasos necesarios para instalar, operar y desinstalar (de ser necesario) el sistema de información.

Y un tercer manual generado en el desarrollo de sistemas es el de operación, que detalla, conforme a los requerimientos del sistema, la forma de utilizar cada uno de sus componentes y funciones.

GLOSARIO DE LA UNIDAD

Documentación

Ciencia del procesamiento de la información. Proporciona información sobre algo con un fin determinado, de ámbito multidisciplinar o interdisciplinar.

Mantenimiento

Según la European Federation of National Maintenance Societies, son todas las acciones que tienen como objetivo mantener un artículo o restaurarlo a un estado en el cual pueda llevar a cabo alguna función requerida.

Manual

Instrumento administrativo que contiene en forma explícita, ordenada y sistemática información sobre objetivos, políticas, atribuciones, organización y procedimientos de los órganos de una institución, así como las instrucciones o acuerdos necesarios para la ejecución del trabajo asignado al personal. El marco de referencia son los objetivos de la institución.

ACTIVIDADES DE APRENDIZAJE

ACTIVIDAD 1

Investiga tres ejemplos de manuales técnicos de sistemas de información disponibles en Internet o en las bibliotecas de las universidades (tesis o trabajos de titulación). Con base en lo anterior, elabora un cuadro comparativo sobre los elementos que integran los distintos manuales investigados. Menciona tus fuentes.

ACTIVIDAD 2

Investiga tres ejemplos de manuales de usuario de sistemas de información disponibles en Internet o en las bibliotecas de las universidades (tesis o trabajos de titulación). Con base en lo anterior, elabora un cuadro sinóptico sobre sus principales características (incluye la forma de redacción de los manuales). Menciona tus fuentes.

ACTIVIDAD 3

Discute con tus compañeros sobre la documentación de sistemas, con base en las siguientes preguntas.

1. ¿Por qué es importante contar con los manuales técnico y de usuario de un sistema de información?
2. ¿Cuáles son los principales problemas que podemos afrontar al no contar con esos manuales?
3. ¿Cuáles son las ventajas de contar con los manuales técnico y de usuario?

CUESTIONARIO DE REFORZAMIENTO

Responde las siguientes preguntas.

1. ¿Qué es el manual técnico?
2. Menciona al menos tres aspectos que considera un manual técnico.
3. ¿Qué es el manual de usuario?
4. ¿Cuáles son las características de un manual de usuario?
5. ¿Qué es un manual operativo?
6. ¿Qué es un plan de mantenimiento?
7. ¿Cómo se relacionan los planes de mantenimiento con el manual técnico?
8. ¿Qué es un plan de mantenimiento multinivel?
9. ¿Cómo es posible realizar el control de los planes de mantenimiento?
10. Menciona algunas recomendaciones a seguir al efectuar alguna actividad de mantenimiento.

LO QUE APRENDÍ

Elabora un cuadro sinóptico sobre los manuales que acompañan a la entrega de un sistema de información. Agrega sus características generales.

EXAMEN DE AUTOEVALUACIÓN

Responde verdadero (V) o falso (F).

1. La documentación es un proceso que se realiza al final del ciclo de vida del sistema. ( )
2. Los sistemas de información deben ser concebidos con la idea de poder ser modificados en un futuro no muy lejano. ( )
3. El manual técnico es un documento que, por su naturaleza, está destinado a ser empleado por ingenieros de *software* o desarrolladores de sistemas, ya que les permite comprender al sistema en su totalidad y realizar alguna modificación. ( )
4. El manual de usuario es un documento dirigido a usuarios finales del sistema en el que se indican los pasos necesarios para modificar cualquiera de sus características. ( )
5. Dentro del manual de usuario, se describen los objetivos del sistema, así como los diferentes menús u opciones que lo conforman. ( )
6. Planificar las acciones de mantenimiento de un sistema ayuda a las empresas a evitar detener sus operaciones a causa de un fallo en el sistema. ( )
7. El manual operativo es de vital importancia, ya que dentro de él se describe el funcionamiento de cada componente y los posibles puntos débiles que afecten la calidad del sistema. ( )
8. Los planes de mantenimiento aseguran que los objetivos y tiempos establecidos para cada actividad sean alcanzados de forma adecuada garantizando la funcionalidad óptima del sistema. ( )

MESOGRAFÍA

BIBLIOGRAFÍA RECOMENDADA

Autor	Capítulo	Páginas
Vasconcelos	5	84-104
Pressman	10	165-180

BIBLIOGRAFÍA BÁSICA (REFERENCIAS)

Vasconcelos, Jorge. *Manual de construcción de programas*. México: 2000, 153 pp.

Pressman, Martínez, Segovia. *Ingeniería del* software. *Un enfoque práctico*, 5.ª ed. España: McGraw-Hill, 2002, 601 pp.

BIBLIOGRAFÍA COMPLEMENTARIA (TEMARIO ANALÍTICO)

1. Bardou, Louis. *Mantenimiento y soporte logístico de los sistemas informáticos.* México: Alfa Omega-Marcombo, 2004, 292 pp.
2. Bochinno, William A. *sistemas de información para la administración, técnicas e instrumentos.* México: Trillas, 2002, 403 pp.
3. Bonsón, Enrique. *Tecnologías inteligentes para la gestión empresarial*. México: Alfa Omega-Rama, 2002, 258 pp.
4. Cornella, Alfons. *Información digital para la empresa, una introducción a los servicios de información electrónica.* México: Alfa Omega-Marcombo, 2004, 196 pp.
5. Lardent, Alberto R. *Sistemas de Información para la gestión empresarial, procedimientos, seguridad y auditoría*. Buenos Aires: Pearson Education-Prentice Hall, 2001. 443 pp.

6. Levine, Guillermo. *Computación y programación moderna*. México: Addison Wesley, 2000. 609 pp.
7. Long, Nancy y Larry Long. *Introducción a las computadoras y a los sistemas de información*, *edición Internet.* México: Prentice Hall, 1999, 416 pp.
8. McLeod, Raymond, Jr. *Sistemas de información gerencial*, 7.ª ed. México: Prentice Hall, 1999, 688 pp.
9. Oz, Effy. *Administración de sistemas de información*, 2.ª ed. México: Thomson-Learning, 2001, 578 pp.
10. Peña R., Baeza-Yates, R. y Rodríguez, J. *Gestión digital de la información. de bits a bibliotecas digitales y la web.* México: Alfa Omega-Rama, 2004, 464 pp.
11. Piattini, Mario, J. Antonio Calvo-Manzano, Joaquín Cervera y Luis Fernández. *Análisis y diseño detallado de aplicaciones informáticas de gestión.* México: Alfa Omega-Rama, 2004, 728 pp.
12. Stair, Ralph M. *Principios de sistemas de información*. 4.ª ed. México: Thomson Learning, 2002, 692 pp.
13. Walker, D. W. *Sistemas e información para la administración.* México: Alfa Omega-Marcombo, 2001, 360 pp.

SITIOS ELECTRÓNICOS

Dirección electrónica	Descripción
http://goo.gl/uacmCy	Curso tutorial sobre documentación de *software*, por Ana Sosa Pintle. Instituto Tecnológico de Puebla.
http://goo.gl/cHE2pp	Ejemplo de manual técnico de *software*, por Patricio Riffo Muñoz de Grupo Marsoft.
http://goo.gl/1PRdL5	Ejemplo de manual de usuario. Universidad Nacional de Córdoba, Argentina.

UNIDAD 8
CASO PRÁCTICO

OBJETIVO ESPECÍFICO

El alumno identificará los conceptos aprendidos a lo largo del curso en el estudio de un caso práctico de un sistema de información.

INTRODUCCIÓN

La aplicación de los conceptos analizados hasta ahora es importante para los futuros profesionistas que desean incursionar en el área de la ingeniería de *software*. Por eso, en la presente unidad, nos enfocaremos al estudio de un caso práctico: el desarrollo de un sistema para laboratorios clínicos de la empresa Sofilab.

En el documento que estudiaremos, aplicaremos la mayor parte de los conceptos revisados hasta ahora. Es importante aclarar que se trata de un trabajo desarrollado para una empresa particular, aunque el caso se presenta dentro de un documento de tesis de licenciatura. Debemos comprender que varios aspectos quedarán omisos por cuestiones de privacidad de la misma empresa, que solamente permitió mostrar de forma limitada la documentación técnica del sistema.

El sistema que se presenta fue instalado a principios del 2000 en varios laboratorios clínicos de hospitales privados y del sector público, como el hospital Juárez de la Ciudad de México, el Ángeles de Interlomas, Laboratorios Jenner, entre otros. En la actualidad, el sistema ya ha cumplido con su ciclo de vida y fue sustituido hace

aproximadamente cinco años por uno nuevo de la misma empresa, cuya estructura y conceptualización base fue el sistema que estudiaremos.

LO QUE SÉ

Escribe un listado de los elementos estudiados en el curso que esperarías encontrar en un caso real de un diseño de un sistema de información.

Temario detallado (12 HORAS)

8. CASO PRÁCTICO

8.1. Solución de un caso práctico, basado en los elementos anteriormente descritos

El caso que estudiaremos a continuación es un trabajo presentado como tesis de titulación para la carrera de Ingeniería en Computación, en la Universidad Nacional Autónoma de México, presentada en abril del 2002 por Marco Polo Manzano Vega y René Montesano Brand. Los autores, en colaboración con Arturo Jallath Coria y la empresa Sofilab, desarrollaron un sistema integral de laboratorio clínico, con el cual se buscaba mejorar la administración de un laboratorio clínico de forma general y adicionalmente obtener de forma directa los resultados de los análisis clínicos de los pacientes hacia el sistema a través de una serie de interfaces diseñadas para conectar el sistema integral con los equipos de análisis clínico.

De manera general, el caso parte desde la presentación de situación predominante que dio origen a la concepción del sistema. A partir de este punto, observaremos paso a

paso cada una de las fases de desarrollo de un sistema de información mencionadas a lo largo de las unidades anteriores.

Como se mencionó en la introducción, por cuestiones de privacidad en la información de la empresa, no se muestra en su totalidad la documentación técnica, pero sí los elementos principales que dan forma al sistema desde su análisis, diseño, desarrollo, implementación y mantenimiento.

En la siguiente liga, tendrás acceso al documento en PDF de la tesis donde se presenta el sistema. Revísalo y resuelve las actividades de aprendizaje asociadas a su contenido.

Tesis de licenciatura “Sistema Integral de Labororatorio Clínico SOFILAB”, presentada por Ing. Marco Polo Manzano Vega & Ing. René Montesano Brand en abril de 2002, Facultad de Ingeniería, UNAM.

RESUMEN DE LA UNIDAD

“El sistema integral de laboratorio clínico Sofilab” es un ejemplo que nos lleva de la mano en la construcción de un sistema de información, partiendo desde la presentación de la problemática para establecer las necesidades que se tenían al comienzo del proyecto. A partir de ese punto, es posible establecer los requerimientos del sistema y plantear el objetivo del sistema, comenzando con la fase de análisis.

En la fase de diseño, se plasma de manera más amplia en el documento la selección del modelo de desarrollo a seguir, tipo de arquitectura, selección del sistema operativo, lenguaje de programación empleado para su desarrollo y las herramientas auxiliares necesarias para completar su construcción. Además, se muestra la planeación con las fechas estipuladas para las entregas y presentación de los avances del sistema, también podemos observar los diagramas de flujo de datos donde se conceptualizará el sistema en su totalidad y el diagrama que conformará la base de datos.

La fase de desarrollo puede observarse a través de los ejemplos de código de programación, de la creación de los procedimientos almacenados en la base de datos, y otros elementos presentados.

De forma general, el documento ofrece un ejemplo de un manual técnico. Parte de la documentación que debe acompañar al sistema, y que se fue generando a lo largo de todo el proceso que dio vida al sistema de información.

Por último, podemos observar la fase de implementación y mantenimiento, que considera desde la instalación del sistema hasta su puesta en marcha. Cabe aclarar que muchos requisitos de implementación fueron establecidos desde la fase de análisis del sistema mismo.

ACTIVIDADES DE APRENDIZAJE

ACTIVIDAD 1

Explica brevemente en un máximo de dos páginas de qué forma se lleva a cabo la fase de análisis del sistema. Resalta los elementos estudiados en las unidades anteriores.

ACTIVIDAD 2

En un mapa conceptual, plasma los elementos que se presentan asociados a las fases de diseño y desarrollo del sistema descrito en el documento. Resalta los elementos abordados en las unidades anteriores.

ACTIVIDAD 3

Discute con tus compañeros sobre las propuestas para el desarrollo del sistema información, con base en las siguientes preguntas.

1. ¿Qué modelo de desarrollo, diferente al empleado en el caso, propondrías como alternativa para el desarrollo del sistema de información? ¿Por qué?
2. ¿Qué elementos adicionales (diagramas de caso, diagramas de flujo, etcétera) emplearías para la fase de diseño del sistema? ¿Por qué?
3. ¿Qué tipo de pruebas aplicarías al sistema antes de implementarlo?

ACTIVIDAD 4

A manera de conclusión del caso, elabora una reflexión en no más de dos páginas sobre la metodología, herramientas y enfoque que hubieras empleado para construir el sistema de información presentado en el caso.

CUESTIONARIO DE REFORZAMIENTO

Responde las siguientes preguntas.

1. ¿Qué es un modelo de desarrollo o modelo de ciclo de vida?
2. ¿Cuál fue el empleado en el caso? Explícalo brevemente.
3. ¿Con base en qué se hace la selección del sistema operativo del caso presentado?
4. Menciona algunas de las herramientas o utilerías adicionales empleadas para la construcción del sistema de información y explica por qué fueron seleccionadas.
5. ¿Por qué es necesaria la planificación de tareas y tiempos en el desarrollo de un sistema de información?
6. Además de los conceptos de ingeniería de *software*, ¿qué otras áreas de la informática están asociadas al sistema de información presentado?
7. ¿Qué elementos de la fase de desarrollo son propias al área de base de datos?
8. ¿En qué elementos del sistema de información son necesarios conceptos del área de telecomunicaciones?
9. ¿Es posible diseñar el sistema bajo el paradigma orientado a objetos?
10. Explica tu respuesta anterior.

LO QUE APRENDÍ

Retoma tu actividad “Lo que sé” y, de acuerdo con la lista que escribiste, elabora un cuadro sinóptico con los elementos que enumeraste y los componentes adicionales que son empleados en el caso para la construcción del sistema de información.

MESOGRAFÍA

BIBLIOGRAFÍA RECOMENDADA

Autor	Capítulo	Páginas
Manzano y Montesano	Todos	5-133

BIBLIOGRAFÍA BÁSICA (REFERENCIAS)

Manzano Vega, Marco P. Montesano y Brand, René. *Sistema integral de laboratorio clínico Sofilab*. Tesis de licenciatura. Facultad de Ingeniería UNAM, México, 2002, 133 pp.

BIBLIOGRAFÍA COMPLEMENTARIA (TEMARIO ANALÍTICO)

1. Bardou, Louis. *Mantenimiento y soporte logístico de los sistemas informáticos.* México: Alfa Omega-Marcombo, 2004, 292 pp.
2. Bochinno, William A. *Sistemas de información para la administración, técnicas e instrumentos.* México: Trillas, 2002.
3. Bonsón, Enrique. *Tecnologías inteligentes para la gestión empresarial*. México: Alfa Omega-Rama, 2002, 258 pp.
4. Cornella, Alfons. *Información digital para la empresa, una introducción a los servicios de información electrónica.* México: Alfa Omega-Marcombo, 2004, 196 pp.
5. Lardent, Alberto R. *Sistemas de información para la gestión empresarial, procedimientos, seguridad y auditoría*. Buenos Aires: Pearson Education-Prentice Hall, 2001.
6. Levine, Guillermo. *Computación y programación moderna*. México: Addison Wesley, 2000.
7. Long, Nancy y Larry Long. *Introducción a las computadoras y a los sistemas de información*, *edición Internet.* México: Prentice Hall, 1999, 416 pp.
8. McLeod, Raymond, Jr. *Sistemas de información gerencial*, 7.ª ed. México: Prentice Hall, 1999, 688 pp.
9. Oz, Effy. *Administración de sistemas de información*, 2.ª ed. México: Thomson-Learning, 2001, 578 pp.
10. Peña R., Baeza-Yates, R. y Rodríguez, J. *Gestión digital de la información. De bits a bibliotecas digitales y la web.* México: Alfa Omega-Rama, 2004, 464 pp.
11. Piattini, M., J. Antonio Calvo-Manzano, Joaquín Cervera y Luis Fernández. *Análisis y diseño detallado de aplicaciones informáticas de gestión.* México: Alfa Omega-Rama, 2004, 728 pp.
12. Stair, Ralph M. *Principios de sistemas de información*, 4.ª ed. México: Thomson Learning, 2002, 692 pp.

13. Walker, D. W. *Sistemas e información para la administración.* México: Alfa Omega-Marcombo, 2001, 360 pp.

SITIOS ELECTRÓNICOS

Dirección electrónica	Descripción
http://goo.gl/aCY3YE	Diseño de sistemas de información, un caso práctico. Por Adolfo Albaladejo. Universidad de Alicante, España.
http://goo.gl/HVg9qj	Diseño de un sistema de información para la gestión y administración de cursos. Por Alfonso Cutro y otros autores. Universidad Nacional del Nordeste, Argentina.

RESPUESTAS A LOS EXÁMENES DE AUTOEVALUACIÓN

DESARROLLO DE SOFTWARE EMPRESARIAL

Unidad 1	Unidad 2	Unidad 3	Unidad 4
1. E	1.F	1. V	1. e
2. I	2.V	2.F	2.g
3. H	3. V	3.V	3.a
4. G	4. F	4. V	4. d
5. B	5. V	5. F	5. b
6. J	6. F	6. F	6. j
7. F	7. F	7. V	7. f
8. A	8. V	8. V	8. c
9. D	9. V	9. V	9. h
10. C	10. F	10.F	10. i
Unidad 5	**Unidad 6**	**Unidad 7**	
1. V	1. F	1. F	
2. V	2. V	2. V	
3. F	3. V	3. V	
4. F	4. F	4. F	
5. F	5. F	5. V	
6. V	6. V	6. V	
7. V	7. V	7. F	
8. F	8. V	8. V	
9. V	9. F		
10. V	10 V		