Universidad Nacional Autónoma de México
Facultad de Contaduría y Administración
Sistema Universidad Abierta y Educación a Distancia

Licenciatura en Informática

Informática III. Análisis y Diseño de Sistemas Estructurado

Apunte electrónico

SUAyED

COLABORADORES

DIRECTOR DE LA FCA
Dr. Juan Alberto Adam Siade

SECRETARIO GENERAL
L.C. y E.F. Leonel Sebastián Chavarría

– – – –

COORDINACIÓN GENERAL
Mtra. Gabriela Montero Montiel
Jefe de la División SUAyED-FCA-UNAM

COORDINACIÓN ACADÉMICA
Mtro. Francisco Hernández Mendoza
FCA-UNAM

– – – –

COAUTORES
Mtro. Gabriel Guevara Gutiérrez
Lic. Karla Ivette Ortega Hernández

CORRECCIÓN DE ESTILO
Mtro. Francisco Vladimir Aceves Gaytan

DISEÑO DE PORTADAS
L.CG. Ricardo Alberto Báez Caballero
Mtra. Marlene Olga Ramírez Chavero
L.DP. Ethel Alejandra Butrón Gutiérrez

DISEÑO EDITORIAL
Mtra. Marlene Olga Ramírez Chavero

OBJETIVO GENERAL

Al finalizar el curso, el alumno aprenderá a desarrollar sistemas utilizando el análisis y diseño apropiados, según el enfoque estructurado.

TEMARIO OFICIAL

(64 horas)

	Horas
1. Introducción	16
2. Análisis de sistemas	28
3. Diseño de sistemas	28
TOTAL	64

INTRODUCCIÓN

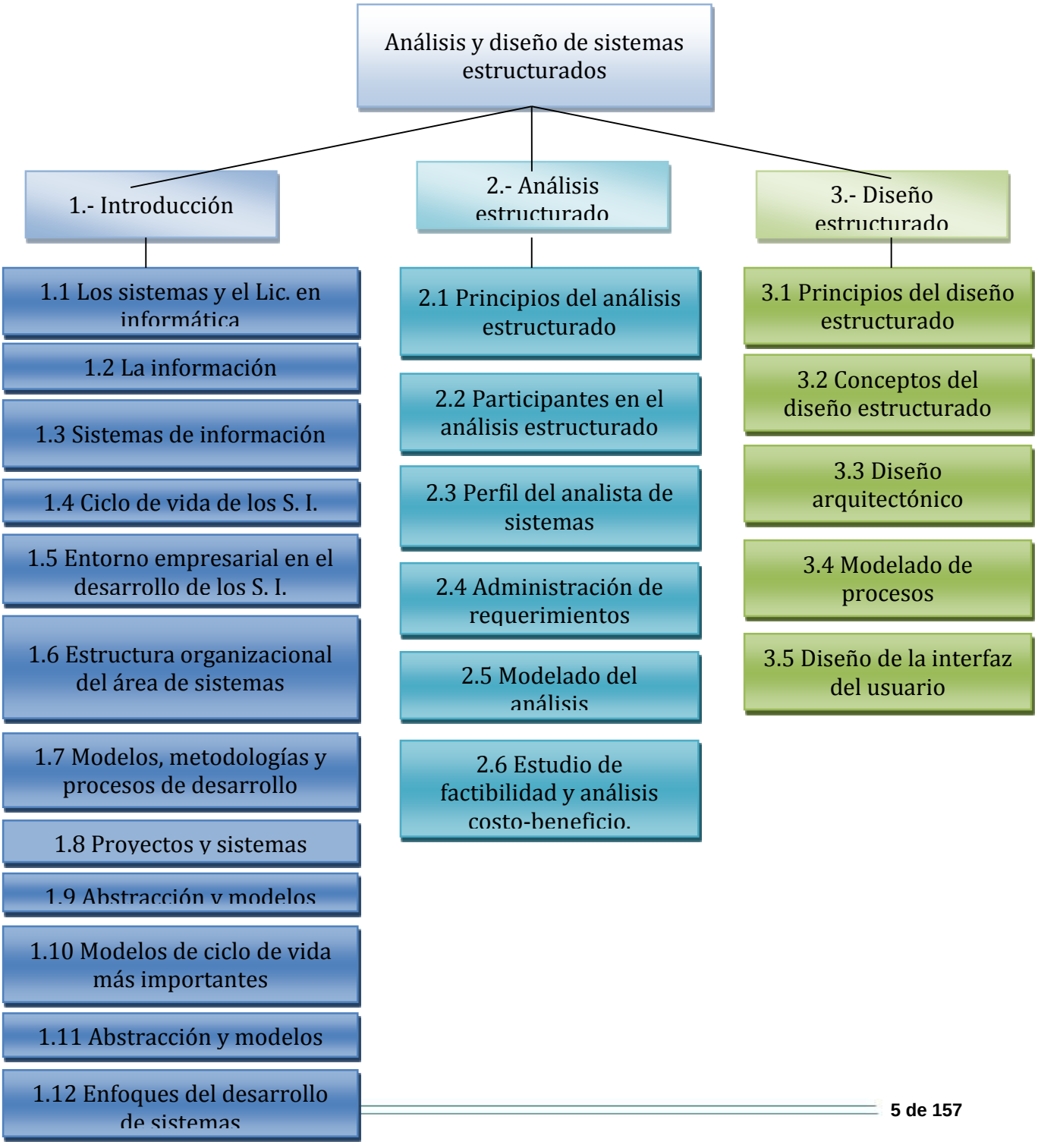
Esta asignatura tiene como propósito señalar los conceptos más relevantes que dan sustento al análisis y diseño de sistemas como etapas del ciclo de vida de los propios sistemas, para que puedas identificarlos. La estructura de la materia se conforma de tres unidades:

1. **Introducción**. Se explican los conceptos que conforman la Teoría General de Sistemas que hoy en día siguen estando vigentes y tienen gran relevancia en el desarrollo de sistemas. Asimismo, se abordan los modelos del ciclo de vida de sistemas que conforman las bases de los procesos de desarrollo de software.

2. **Análisis Estructurado**. Se explican los conceptos que intervienen en la administración de requerimientos y en el análisis de sistemas.

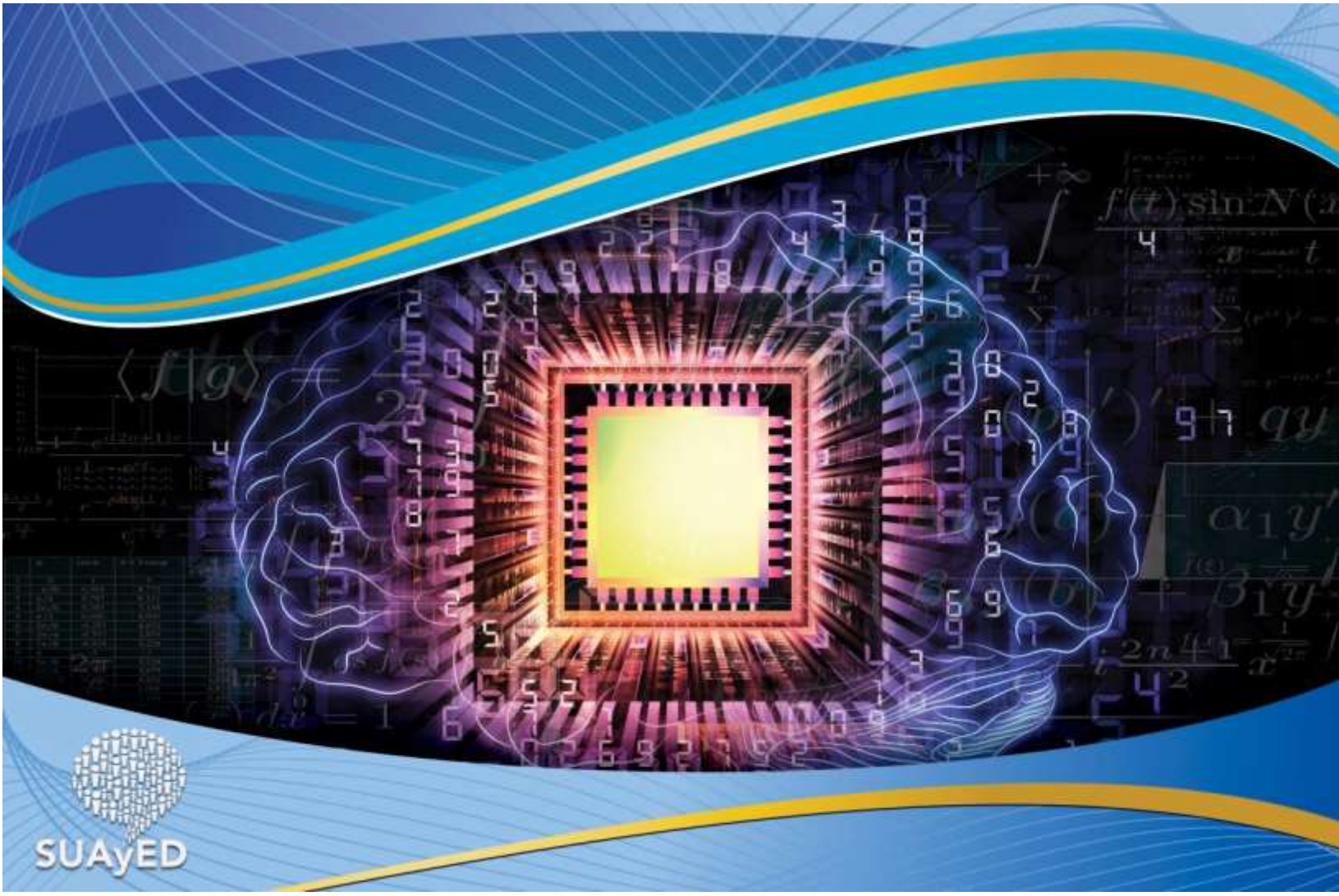
3. **Diseño Estructurado**. El contenido de esta asignatura es fundamental para tu formación como Licenciado en Informática; la asignatura de *Análisis y diseño estructurado de sistemas* conforma la base para asignaturas que cursarás en el mismo semestre, así como para futuros semestres, por tal razón, necesitas poner especial atención en los conceptos y procedimientos que aquí se explicarán.

ESTRUCTURA CONCEPTUAL



UNIDAD 1

INTRODUCCIÓN



OBJETIVO PARTICULAR

Al finalizar el estudio de la unidad, el alumno identificará el concepto de sistemas, los modelos y las metodologías de proceso.

TEMARIO DETALLADO
(16 horas)

1. Introducción

- 1.1. Los sistemas en la actividad profesional del Licenciado en Informática
- 1.2. La información
  - 1.2.1. Definición de información
  - 1.2.2. Propiedades o atributos de la información
  - 1.2.3. Fuentes de la información
  - 1.2.4. Teoría General de Sistemas
- 1.3. Sistemas de Información
  - 1.3.1. Definición de Sistema de Información
  - 1.3.2. Componentes de un Sistema de Información
  - 1.3.3. Clasificación de los Sistemas de Información
  - 1.3.4. Sistemas de Información y Sistemas Informáticos
- 1.4. Ciclo de vida de los Sistemas de Información
  - 1.4.1. Análisis
  - 1.4.2. Diseño
  - 1.4.3. Implementación
  - 1.4.4. Pruebas
  - 1.4.5. Implantación
  - 1.4.6. Mantenimiento

INTRODUCCIÓN

Los sistemas y la información son los elementos básicos que conforman a los llamados “sistemas de información”, por tal motivo su definición es de suma importancia para poder entender el concepto en sí.

Existen varios tipos de *sistemas de información* cuya clasificación está en función de la forma en que procesan la información y el nivel jerárquico de la organización que hace uso de ella.

Para desarrollar un sistema de información se hace uso de las etapas que conforman el ciclo de vida de los sistemas (análisis, diseño, implementación, pruebas, implantación, mantenimiento), sin embargo, existen diversas formas de aplicar estas etapas, las cuales son denominadas “modelos”.

En esta unidad se revisará la información pertinente, es decir, los elementos conceptuales necesarios, para comprender el área de desarrollo de los sistemas.

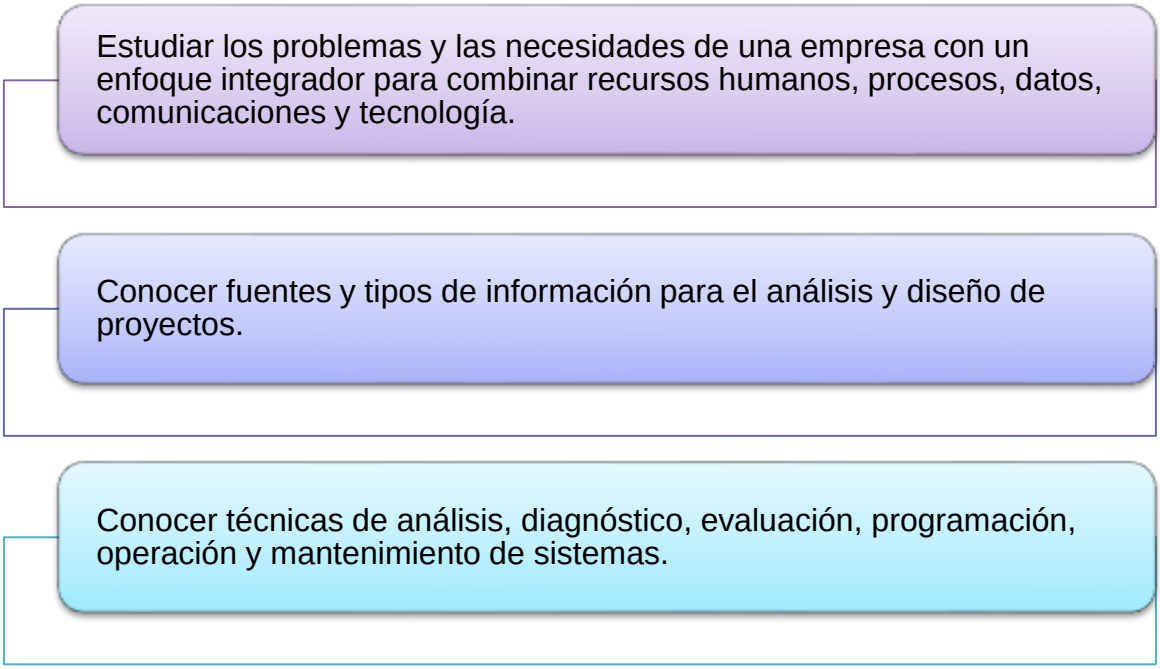
1.1. Los sistemas en la actividad profesional del licenciado en informática

Sin lugar a dudas, en el presente Siglo XXI el concepto de *sistema* y su comprensión tiene una gran importancia para cualquier ciencia o disciplina.

La Licenciatura en Informática tiene como objetivo formar profesionales capaces de diseñar e implantar soluciones basadas en sistemas de información, que faciliten la toma de decisiones y agilicen las operaciones propias de una organización. Es el experto que planea, dirige y controla el desarrollo y funcionamiento óptimo de los centros de información y los recursos informáticos, mediante la aplicación de las mejores técnicas y metodologías de evaluación, selección e implantación de su arquitectura, así como del desarrollo de sistemas administrativos de información.

Los sistemas de información son una herramienta básica para transformar las necesidades de información y de empresa en soluciones técnicas basadas en computadoras capaces de almacenar y procesar enormes cantidades de datos a una gran velocidad y de forma precisa, suministrando de inmediato la información relevante a los directivos.

Para ello el Licenciado en Informática debe:



1.2. La información

Es preciso comentar que la finalidad del estudio de estos conceptos es que tú puedas identificar el concepto de “información”, así como sus atributos y propiedades, además de sus diferentes orígenes para poder obtenerla a partir de la revisión de la Teoría General de Sistemas. Una vez realizado esto, podrás elaborar tu propia definición de un “sistema de información”, con sus componentes y clasificación, al igual que la diferencia entre un *sistema de información* y un *sistema informático*. Revisemos el concepto de información.

1.2.1. Definición de información

A continuación unas cuantas definiciones de distintos autores:

> Por Información se entienden datos a los que se les ha dado una forma que tiene sentido y es útil para los humanos. En contraste, los datos son secuencias de hechos en bruto y representan eventos que ocurren en las organizaciones o en el entorno físico, antes de ser organizados y ordenados en una forma que las personas puedan entender y usar." (Laudon y Laudon, 2004, p. 8)

> *Información*: Es un dato que ha sido manipulado, con lo que resulta de utilidad para alguien y debe tener valor, o en caso contrario no sería un dato. […] *Dato*: Es una colección de hechos considerados de forma aislada que portan un significado pero en general no son de utilidad por sí solos. (Whitten, Barlow & Bentley, 2003)

> La *información* es un conjunto de datos cuya relación entre sí da un significado a un ente (persona u organización) para tomar una decisión. El valor de la información es totalmente subjetivo, ya que depende de la persona u organización y de sus características. (Definición ABC, aquí)

> *Dato* proviene del idioma latín, del vocablo "*datum*", y se refiere a una representación mediante símbolos numéricos, alfabéticos o de otra clase de la característica de algo. (Definición ABC, aquí)

1.2.2. Propiedades o atributos de la información

La información debe de cumplir con ciertas propiedades o características para que se le pueda considerar “información”. Estas características son las siguientes:

Propiedad	Descripción
Exacta	La información debe ser precisa y libre de errores.
Completa	La información debe contener todos aquellos hechos que pudieran ser importantes.
Económica	El costo que se debe incurrir para obtener la información debe ser menor que el beneficio proporcionado a la organización.
Confiable	La información debe garantizar tanto la calidad de los datos utilizados, como la de sus fuentes de información.
Relevante	La información debe ser útil para la toma de decisiones. Evitar todos aquellos hechos que sean superfluos o que no aporten algún valor.
Detallada	La información debe presentar el nivel de detalle indicado para la decisión a la que se destina. Se debe proporcionar con la presentación y el formato adecuados para que resulte sencilla y fácil de manejar.
Oportuna	La información debe ser entregada a la persona y en el momento que la necesita para poder tomar la decisión.
Verificable	La información debe poder ser contrastada y comprobada.

Propiedades de la información (Gómez y Suárez, 2007, pp. 5-6)

1.2.3. Fuentes de la información

Son todos los recursos que contienen datos formales, informales, escritos, orales o multimedia.

Tipo de fuente	Descripción
Primaria	Contienen información original, que ha sido publicada por primera vez y que no ha sido filtrada, interpretada o evaluada por nadie más.
Secundaria	Es información resultado del análisis, síntesis, interpretación o evaluación de una fuente primaria.

No obstante, la información puede tener diferentes orígenes y formatos a la vez; en el caso del desarrollo de sistemas de información, que es el que nos compete, la información que requiere un sistema tiene sus fuentes en:

Fuente	Descripción
Personas	Las personas son la fuente principal para el analista de sistemas, ya que de ellas obtiene la información necesaria para la construcción del sistema o para la entrada de datos.
Documentos	Los documentos se refieren a la información contenida en un medio físico o electrónico, dentro o fuera de la organización, que describen de manera parcial o completa cómo funciona. Ejemplos: Manual de Procedimientos, Manual de Organización, Políticas, Normas, Informes, Reportes, Expedientes, etcétera.
Sistemas	Actualmente, la información puede obtenerse de forma manual o automática y en esta última los sistemas (informáticos) son las principales fuentes de información para las organizaciones.

Pasemos ahora al estudio de la Teoría general de sistemas.

1.2.4. Teoría general de sistemas

La Teoría General de Sistemas, por sus siglas TGS, fue desarrollada en 1925 por Ludwig von Bertalanffy. Una de sus ideas más sobresalientes indica que no es suficiente analizar un elemento en forma aislada sino que es indispensable analizar al elemento junto con los demás que se encuentra relacionado. Con esto se establece que el comportamiento de un elemento del sistema afectará, en menor o mayor grado, a aquellos elementos con los que se encuentra relacionado.

Ludwig von Bertalanffy
(http://bit.ly/L3hHcq, recuperado el 24/03/12)

La TGS es un conjunto de modelos, principios y leyes válidos para cualquier tipo de sistema no importando la naturaleza de sus elementos ni las relaciones entre ellos. Como la TGS es general, su espectro es amplio, así que no es de extrañarse que veamos su estudio en la Biología, en la Administración, en la Informática, entre otras áreas.

La importancia que tiene la TGS en el Análisis y Diseño de Sistemas está dada porque, en cualquier tipo de proyecto que tenga como objetivo la creación de un sistema de información, se hacen indispensables la integración de conocimientos y la especialización con el fin de reducir tiempos y costos, pero, sobre todo, se hace indispensable la comprensión del funcionamiento del todo, entiéndase el sistema. Si necesitamos un enfoque reduccionista, al analizar sólo una parte del sistema

tendríamos una visión limitada del problema y, como consecuencia, no identificaríamos las causas que lo originan.

Dentro de los conceptos más destacados de esta teoría tenemos:

Sinergia	•El conjunto de elementos tiene más valor que la simple suma de las partes.
Entropía	•Es el desgaste natural que sufre el sistema por los cambios que se presentan en el ambiente. Sólo existe en modelos teóricos y ambientes estáticos. Todo ambiente provoca cambios en los sistemas que rodea, esto trae como consecuencia que el sistema tenga que responder a tales cambios de alguna manera.
Isomorfismo	•Define un conjunto de similitudes estructurales que pueden compartir algunos sistemas.
Cibernética	•Teoría de los sistemas de control basada en la comunicación entre el sistema y el ambiente.
Equifinalidad	•Se refiere al hecho de que un sistema vivo, a partir de distintas condiciones iniciales y por distintos caminos, llega a un mismo estado final. El fin se refiere a la mantención de un estado de equilibrio fluyente.

1.3. Sistemas de información

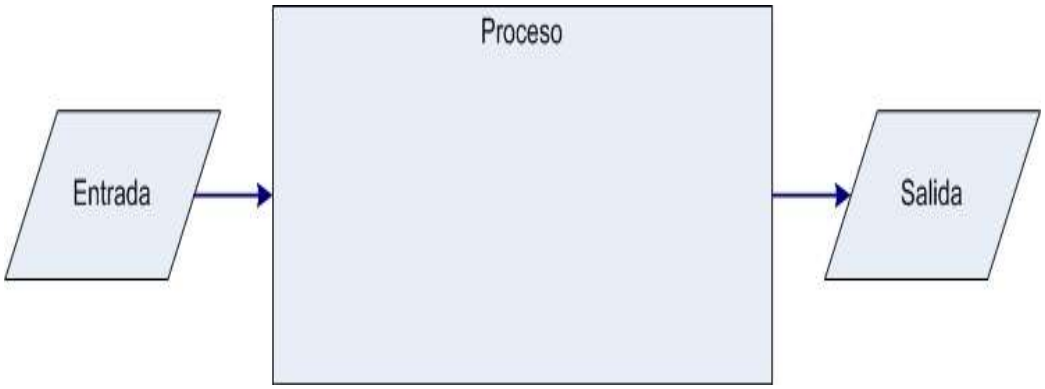
Un sistema es un conjunto de actividades o componentes relacionados recíprocamente para contribuir a la realización de metas planeadas previamente; por su parte Mora y Molino (1993) también exponen que “es un conjunto de elementos y procedimientos íntimamente relacionados que tienen como propósitos el logro de un objetivo determinado” (p. 27).

13.1. Definición de sistema de información

Laudon y Laudon (2004) definen un “sistema de información” como “componentes interrelacionados que colaboran para reunir, procesar, almacenar y distribuir información que apoya la toma de decisiones, la coordinación, el control, el análisis y la visualización en una organización” (p. 8).

Un sistema, en su concepción más sencilla, es como una *caja negra* que recibe una entrada, la cual es procesada para generar una salida. Bajo esta concepción, los elementos básicos de un sistema son:

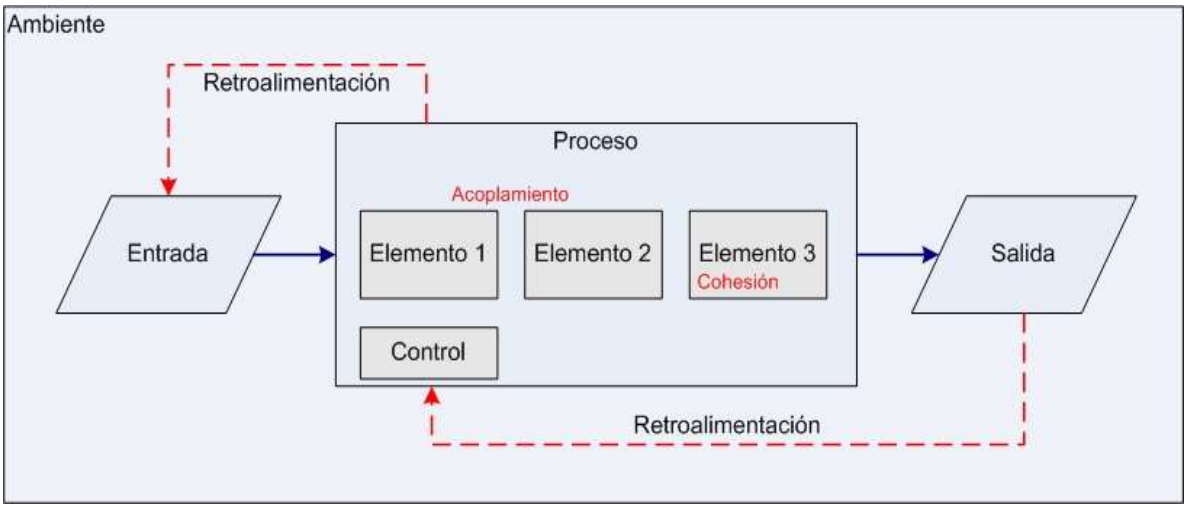
Entradas (ejemplos):	Datos generales del cliente: nombre, dirección, tipo de cliente, etc.
	Políticas de créditos: límite de crédito, plazo de pago, etc.
	Facturas (interface automático).
	Pagos, depuraciones, etc.
Proceso (ejemplos):	Cálculo de antigüedad de saldos.
	Cálculo de intereses moratorios.
	Cálculo del saldo de un cliente.
Salidas (ejemplos):	Reporte de pagos.
	Estados de cuenta.
	Consultas de saldos en pantalla de una terminal.



Vista de la caja negra de un sistema

Otra concepción de *sistema* nos indica que es un conjunto de elementos.[1]

[1] Otras palabras que se pueden emplear en lugar de elementos son: 'partes', 'componentes' y 'módulos'. interrelacionados e interdependientes entre sí que buscan un objetivo común. El hecho de que los elementos busquen un objetivo común no excluye la capacidad de que cada elemento tenga un objetivo en particular.



Vista de la caja blanca de un sistema

El *ambiente* es el entorno que rodea al sistema y, por ende, lo afecta directamente con algún tipo de estímulo. Teóricamente, cada sistema cuenta con una “frontera” o “límite” que lo distingue del ambiente en el que se desenvuelve; por ejemplo, la frontera del cuerpo humano es la piel.

La retroalimentación es una actividad de autorregulación, la cual supervisa que las salidas que genera el sistema cumplan con ciertas características basadas en las entradas y su proceso de transformación.

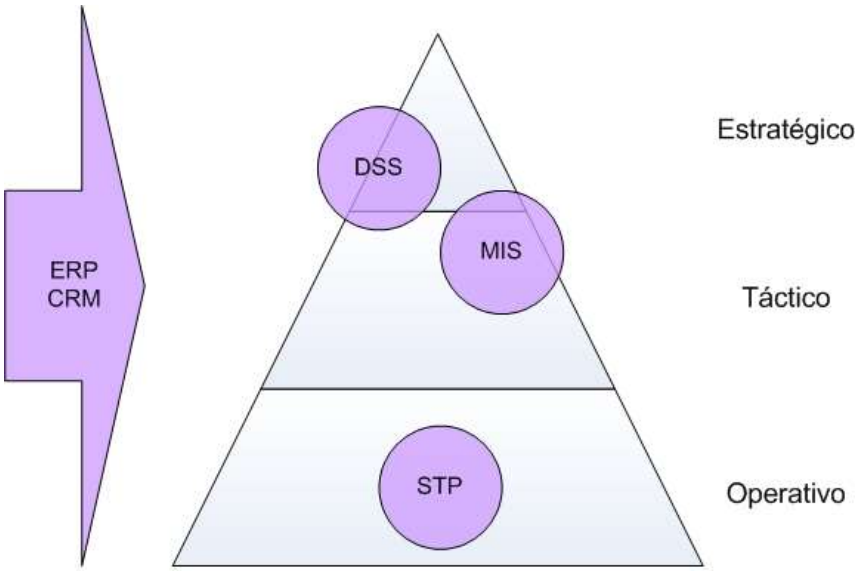
1.3.2. Componentes de un Sistema de Información

En este tipo de sistema se presentan tres componentes más:

- Proceso Es el conjunto de actividades por realizar para generar un producto (bien o servicio). En el caso de los *sistemas de información* consiste en algún tipo de información para la toma de decisiones; también es conocido como “procedimiento”.
- Persona Es el individuo que tiene la responsabilidad de crear, modificar, destruir y verificar un insumo que entra al sistema o que es procesado por el sistema.
- Base de datos Es un conjunto de datos almacenado en cierto orden para que, posteriormente, puedan ser recuperados.

1.3.3. Clasificación de los Sistemas de Información

Básicamente, en cualquier organización se distinguen tres niveles para la toma de decisiones (de nivel inferior a nivel superior): operativo, táctico y estratégico; de una manera similar podemos clasificar la información. Dependiendo del nivel del que se trate, las personas necesitarán de información operativa (información muy detallada), táctica (información resumida) o estratégica (información sintetizada).



Pirámide del Nivel de Tipo de Decisiones y Sistemas de Información

Los diferentes tipos de sistemas de información se ubican en alguno de los niveles o en varios de ellos:

TPS: (*Transaction Proccessing System* – Sistema de Procesamiento de Transacciones)	•Este sistema tiene como objetivo recoger datos básicos en las operaciones organizacionales, enfocándose principalmente en las áreas de contabilidad, ventas, compras y nómina.
MIS: (*Management Information System* – Sistema de Información Administrativa)	•Este sistema genera informes que permiten a los directivos mejorar el control de la ejecución de las actividades en la organización.
DSS: (*Decision Support System* – Sistema de Apoyo a las Decisiones)	•Este sistema tiene como objetivo apoyar y asistir en el nivel estratégico al proceso de toma de decisiones, creando escenarios o generando alternativas.
ERP: (*Enterprise Resource Planning* – Planeación de Recursos Empresariales)	•Este sistema agiliza la administración de los recursos materiales y humanos en la organización, integrando la información de las diferentes áreas con un enfoque orientado a procesos.
CRM: (*Customer Relationship Managment* – Administración de Relaciones con el Cliente)	•Este sistema tiene como objetivo proporcionar información completa sobre los clientes, la cual incluye qué compran, cuándo compran, qué buscan y qué comentarios tienen sobre la organización.

1.3.4. Sistemas de Información y Sistemas Informáticos

El propósito de un sistema de información y uno informático es el mismo, la diferencia radica en que el sistema informático hace uso de ciertos elementos característicos para lograr el objetivo:

Hardware	Software
•Es el equipo de cómputo y telecomunicaciones utilizado para procesar los datos.	•Son los programas de la computadora donde encontramos sistemas operativos, lenguajes de programación, sistemas manejadores de base datos, procesadores de textos, generadores de presentaciones, etc.

Los sistemas informáticos son un subtipo de los sistemas de información, por lo que no se incurre en un error al hablar de la utilización de algún tipo de hardware o software en el contexto de los sistemas de información.

Conjuntos: Sistema de Información y Sistema Informático

1.4. Ciclo de vida de los sistemas de información

La concepción de *sistemas* tiene su origen en las ciencias naturales por tratar de describir un ser vivo como un conjunto de partes, el cual tiene un ciclo de vida (*nace, crece, se reproduce y muere*). Los sistemas de información tienen su propio ciclo: *análisis, diseño, implementación, pruebas, implantación y mantenimiento*. Dependiendo del autor que se consulte, las etapas o fases pueden ser más o pueden ser menos, también pueden integrarse en una sola o cambiar de nombre, aun así, en esencia tienen el mismo propósito.

Es necesario recalcar que el ciclo de vida de los sistemas es un modelo teórico que tiene como propósito agrupar las actividades en fases por realizar para construir un sistema. Por otro lado, el modelo del "Ciclo de Vida de los Sistemas" no indica la forma o secuencia en que se tienen que aplicar las fases. A continuación se presenta una breve descripción de las fases antes mencionadas:

1.4.1. Análisis

Esta fase contesta la pregunta ¿qué? y es una descripción de las causas, necesidades y problemas existentes, por lo que se puede observar y modelar cómo funciona la organización o sistema actual. Otro nombre que recibe esta fase es el de "requerimientos" y sus principales actividades son las siguientes:

- Diagnosticar la necesidad o problema a resolver
- Identificar a los participantes, así como sus necesidades
- Establecer los objetivos que persigue el sistema
- Establecer los alcances y exclusiones
- Crear un modelo de dominio o negocio con su respectiva especificación, describiendo la situación actual

1.4.2. Diseño

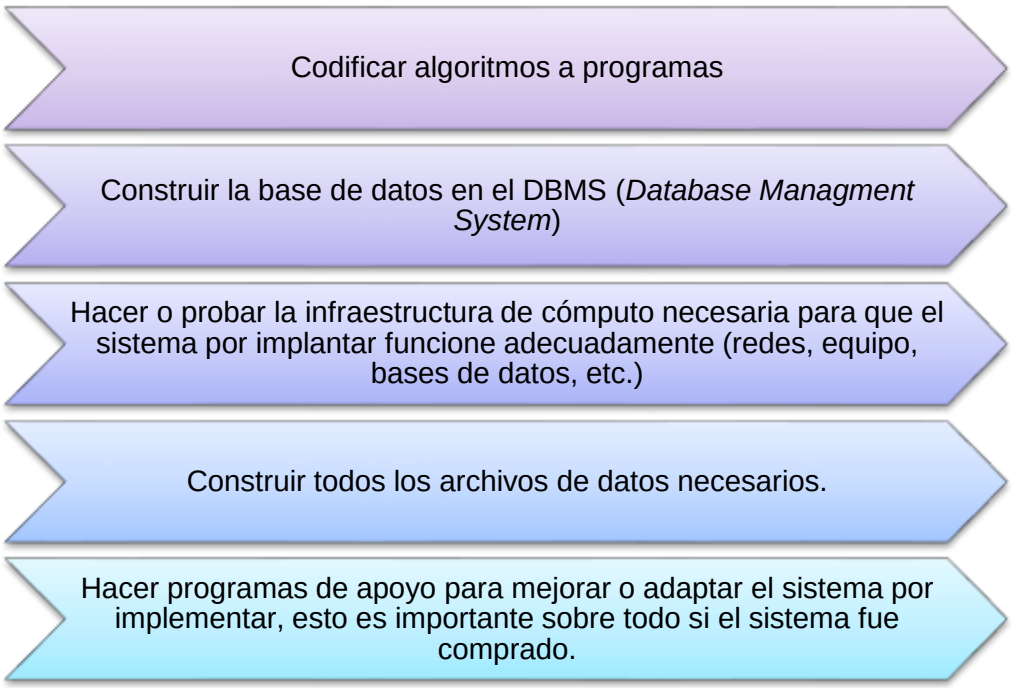
Esta fase contesta a la pregunta ¿cómo?, no hay que confundir el cómo de la fase de análisis, donde se investiga cómo se hacen las cosas; éste se refiere a cómo voy a dar solución a las necesidades y/o problemas. Sus actividades principales son:

- Diseñar procesos y procedimientos
- Diseñar algoritmos
- Diseñar la base de datos
- Diseñar la distribución de los componentes

Regularmente los puntos anteriores reciben el nombre de arquitectura de software o simplemente arquitectura. La arquitectura describe con diferentes modelos cómo se va a construir la solución: “una arquitectura de software define la estructura general de un sistema y varía de acuerdo con el tipo de sistema a desarrollarse” (Weitzenfeld, 2004, p.36).

1.4.3. Implementación

Esta fase se encarga de construir los componentes del software que le darán funcionalidad al sistema. Otros nombres que recibe esta fase son los de “desarrollo”, “programación” o “construcción” y sus actividades principales son:



1.4.4. Pruebas

Esta fase se encarga de asegurar que la calidad del software sea la correcta: por un lado, se verifica que el software se construya de manera correcta, es decir, que se apegue a los estándares establecidos y, por el otro, se valida que el software que se construya sea el correcto, es decir, que satisfaga las necesidades por las cuales se concibió el sistema. Otros nombres que recibe esta fase son los de “verificación”, “validación” o “evaluación”.

1.4.5. Implantación

Algunos lo consideran desarrollo de sistemas lo cual es erróneo porque el desarrollo está en todo el ciclo de vida. Esta fase se encarga de poner en marcha el software construido y capacitar a las personas involucradas para su uso cuyas fases son:

- Instalación del nuevo sistema
- Entrega del sistema para su puesta en marcha

1.4.6. Mantenimiento

Esta fase se encarga de darle mantenimiento al sistema en caso de que surja alguna falla no detectada o se requiera extender, adecuar o mejorar el sistema. Estas formas de mantenimiento reciben los nombres de “Mantenimiento Correctivo”, “Mantenimiento Preventivo” y “Mantenimiento Adaptativo”.

(http://bit.ly/Kay087, recuperado el 24/03/12)

1.5. Entorno empresarial de desarrollo de los sistemas de información

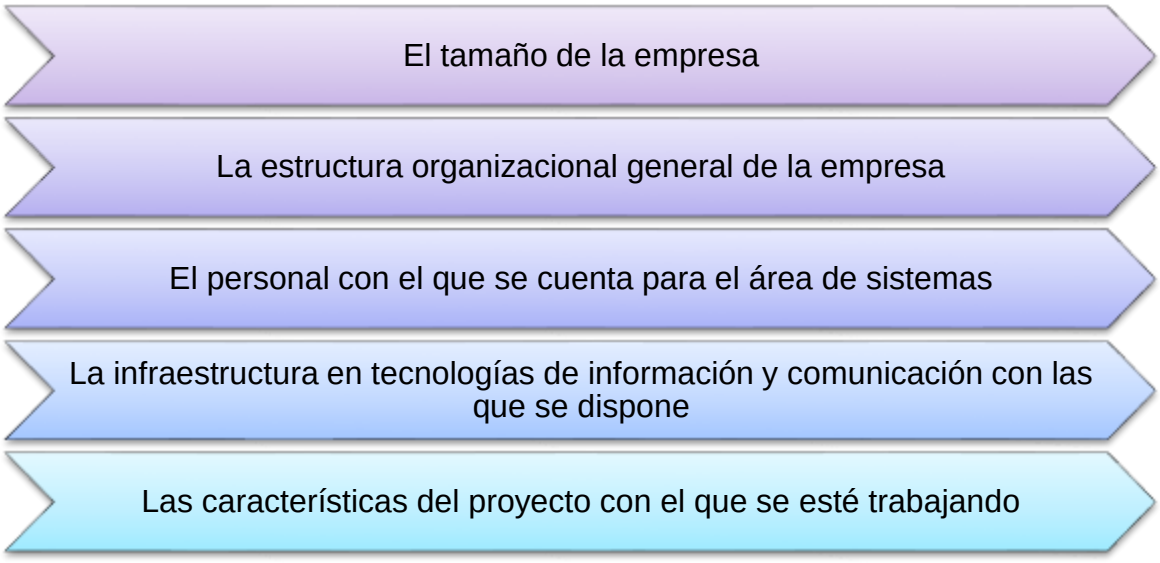
El ambiente en el que se desenvuelve el equipo de desarrollo que va a construir el sistema es altamente cambiante, ya sea que se trate del ambiente interno o del ambiente externo de la organización. Una de las constantes en las organizaciones que se dedican al desarrollo de sistemas es que están orientadas a los procesos y, por consecuencia, están orientadas a sus clientes.

Las empresas no se deben preocupar por crear en primer lugar una estructura organizacional, sino por diseñar y establecer un proceso que permita construir las características de su producto (bien o servicio) para que dichas características satisfagan las necesidades de sus clientes. Una vez definido el proceso, ahora sí, deben preocuparse por crear la estructura organizacional que permita ejecutar el proceso.

Tomando en cuenta lo anterior, se hace indispensable que la persona encargada del área de sistemas esté al tanto de los procesos críticos de la organización para la que está trabajando. Si lo hace, tendrá definidas claramente las metas que debe perseguir el área de sistemas, las cuales buscarán siempre agilizar y optimizar esos procesos.

1.6. Estructura organizacional del área de sistemas

Por lo regular se tiende a crear un departamento por cada una de las fases del ciclo de vida de los sistemas; sin embargo, la definición de la estructura organizacional depende de factores como:



Con base en lo anterior, lo importante es identificar las actividades críticas que nos ayudan a construir el sistema; de esa manera, identificamos la cantidad y el perfil de las personas que necesitamos para ejecutar las actividades y, con eso, se establece la estructura organizacional más adecuada para no perder el control del proyecto y de las personas.

1.6.1. Departamentos Involucrados

El Área o Departamento de Sistemas se ha convertido en uno de motores críticos de las organizaciones, por ello, no es de extrañarse que el departamento tenga relación con cada una de las demás áreas existentes. Tampoco es extraño que el responsable de esta área sea convocado en las decisiones estratégicas que se toman debido a que la empresa es un sistema social. Por lo general existen las siguientes áreas:

1.6.2. Funciones y responsabilidades de cada departamento

Departamento o área de Producción y Control:

- Evalúa las aplicaciones con base en las necesidades de información.
- Usa las técnicas de construcción de sistemas de información orientadas netamente a la productividad del personal y a la satisfacción plena del usuario.
- Construye equipos de trabajo con la participación del usuario y del personal técnico de acuerdo con metodologías establecidas.
- Diseño de reportes, la evaluación de los trabajos efectuados por el personal de los departamentos usuarios, la supervisión de cambios de equipo la preparación de presupuesto en el área de cómputo.
- Mantiene comunicados a los usuarios y a sus colaboradores de los avances, atrasos y problemas que se presentan rutinariamente y cuando sea necesario a través de medios establecidos formalmente, como el uso de correo electrónico, mensajes relámpagos o flash.
- Mantiene programas de capacitación para el personal técnico y usuarios.

Departamento o área de Análisis de Sistemas:

- Establece un flujo de información eficiente a través de toda la organización.
- Realiza el estudio y la proposición de soluciones de los problemas, planteando diferentes alternativas.
- Puede estar agrupado en equipos cuyas funciones son coordinadas por analistas líder o jefes de análisis. Existen diferentes títulos de analistas: Analista Junior, Aprendiz de Sistemas y Analista Senior que indican diferentes grados de experiencia, entrenamiento y educación.

Departamento o área de Programación:

- Elabora los programas que se ejecutan en las computadoras, modifica los existentes y vigila que todos los procesos se ejecuten correctamente.
- Toma las especificaciones de los sistemas realizados por los analistas.
- Genera programas alternos para mejorar o adaptarlo al software comprado.
- Los programadores pueden clasificarse en: "Programadores junior" o "Aprendices de Programación", "Programadores Senior"
- Selecciona, modifica y mantiene el complejo software del sistema operativo.

Departamento o área de Implementación se encarga de:

- Implantar nuevas aplicaciones garantizando tanto su calidad como su adecuación a las necesidades de los usuarios.
- Coordinar con las áreas de sistemas y usuarios la implantación de las aplicaciones.
- Diseñar los planes de calidad de las aplicaciones y garantizar su cumplimiento.
- Validar los nuevos procedimientos y políticas de las implementaciones de los proyectos liberados.
- Probar los productos y servicios por implementar antes de ser liberados al usuario final.
- Elaborar conjuntamente con el área de Programación o Desarrollo, los planes de capacitación de los nuevos usuarios.
- Coordinar la presentación de las nuevas aplicaciones a los usuarios.
- Supervisar el cumplimiento de los sistemas con la normatividad establecida.

Departamento o área de Soporte Técnico se encarga de:

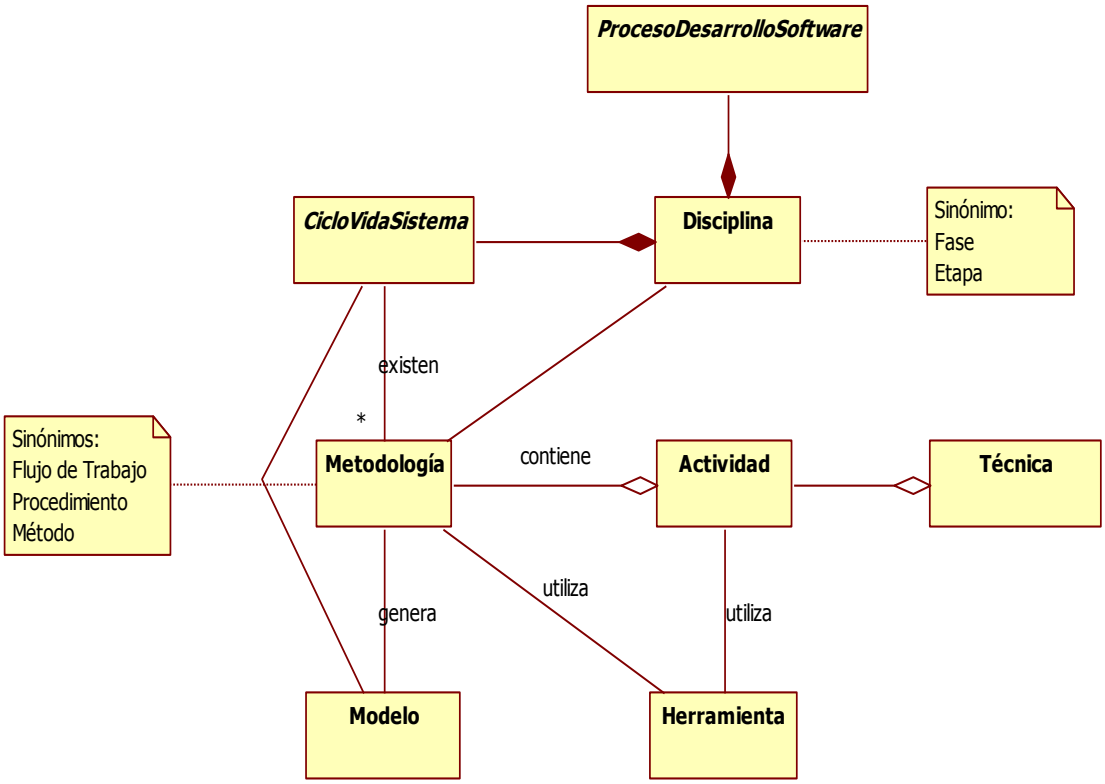
Departamento o área de Soporte Técnico se encarga de:

- La gestión del hardware y del software dentro de las instalaciones.
- Planificar la modificación e instalación de nuevo software y hardware.
- Evaluar los nuevos paquetes de software y nuevos productos de hardware.
- Dar el soporte técnico necesario para el desarrollo de nuevos proyectos, evaluando el impacto de los nuevos proyectos en el sistema instalado.
- Asegurar la disponibilidad del sistema, y la coordinación necesaria para la resolución de los problemas técnicos en su área.
- Realizar la coordinación con los técnicos del proveedor con el fin de resolver los problemas técnicos y garantizar la instalación de los productos.
- Proponer las notas técnicas y recomendaciones para el uso óptimo de los sistemas instalados.
- Participar en el diseño de la Arquitectura de Sistemas.

1.7. Modelos, metodologías y procesos de desarrollo

Al revisar diversas fuentes se puede presentar una gran confusión entre los conceptos de *modelo*, *metodología*, *método*, *ciclo de vida* y *proceso de desarrollo* al utilizarse indistintamente el uno del otro.

En la siguiente figura se muestran las relaciones que existen entre los conceptos mencionados para evitar posibles confusiones.



Asociación de conceptos: Ciclo de Vida, Proceso de Desarrollo, Metodología y Modelo

Conceptos Asociaciones	Descripción
Ciclo de vida de un sistema	Es el conjunto de fases que se tienen que llevar a cabo para construir un sistema de información.
Proceso de desarrollo de software	Este proceso indica qué se tiene que hacer, cómo se tiene que hacer, cuándo se tiene que hacer y quién lo tiene hacer para construir un software con calidad.

Disciplina	Se le conoce como disciplina en el proceso de desarrollo de software y como fase en el ciclo de vida de sistemas. Como tal, es un área de conocimientos y actividades por realizar para construir una parte del sistema de información.
Metodología	No hay un consenso respecto al concepto de metodología. Maddison (1983; 45) define la metodología como un "conjunto de filosofías, fases, procedimientos, reglas, técnicas, herramientas, documentación y aspectos de formación para los desarrolladores de sistemas de información". En (Marylin y Loaiza, 2008) La Metodología especifica: ▪ ¿Cómo se debe dividir un proyecto en etapas? ▪ ¿Qué actividades se llevan a cabo en cada etapa? ▪ ¿Qué salidas se producen y cuándo se deben producir? ▪ ¿Qué restricciones se deben aplicar? ▪ ¿Qué herramientas se van a utilizar? ▪ ¿Cómo administra el proyecto?
Actividad	Es una acción por realizar para generar, actualizar o revisar algún producto de trabajo, como documentos, modelos o software.
Técnica	Dice cómo realizar una actividad, apoyándose de herramientas específicas para cada caso.

Herramienta	Indican con qué contamos para resolver una actividad específica. Hoy día estas herramientas reciben el nombre de CASE (*Computer Assisted Software Engenering*).
Ciclo de vida de un sistema ⇔ Disciplina	El ciclo de vida de un sistema está compuesto de una serie de disciplinas o fases. Las más comunes son el análisis, el diseño, la implementación, las pruebas, la implantación y el mantenimiento.
Proceso de Desarrollo de un Software ⇔ Disciplina	Un proceso de desarrollo de software está compuesto de una serie de disciplinas o fases. Además de las fases que incluye el ciclo de vida de los sistemas, están la administración de proyectos; la administración de la configuración y cambios, y el ambiente.
Ciclo de vida de un sistema ⇔ Metodología	El ciclo de vida de un sistema tiene una o más metodologías. Nota: ver concepto de metodología.
Disciplina ⇔ Metodología	Una disciplina puede ejecutarse bajo diferentes metodologías. Las metodologías dependerán del autor y podrán tener dos enfoques: estructurado u orientado a objetos.
Ciclo de vida de un sistema ⇔ Modelo	El ciclo de vida de un sistema tiene uno o más modelos. Es decir, existen diversas formas de aplicar o secuenciar cada una de las disciplinas o fases. Nota: ver Tema 1.7.3.

Metodología ⇔ Modelo	Una metodología genera gran cantidad de modelos que representan el aspecto estático y dinámico del sistema.
Metodología ⇔ Actividad	Una metodología está compuesta de varias actividades.
Actividad ⇔ Técnica	Una actividad está compuesta de varias técnicas.

Asociación de conceptos

1.8. Proyectos y sistemas

Es muy común confundir el concepto de “proyecto” con el de “sistema”; sin embargo, existe una diferencia. El PMBoK (*Project Management Body of Knowledge*), del PMI (*Project Management Institute*), define al *proyecto* como un esfuerzo temporal emprendido para crear un producto, servicio o resultado único. Esto significa que se puede emprender un proyecto para crear un nuevo coche, construir una casa o, como en este caso, crear un sistema.

PMBoK (http://www.projectsmart.co.uk/img/pmi.png, recuperado el 24/0/12)

Recordemos la definición de *sistema*: es un conjunto de elementos interrelacionados entre sí para conseguir un objetivo común; entonces, este sistema se va a construir a través de la administración de un proyecto.

Aquí es necesario destacar que la confusión entre proyecto y sistema surge porque algunas de las actividades que se llevan a cabo en la administración de proyectos y el ciclo de vida de los sistemas son iguales o similares; por ejemplo, la obtención de requerimientos es una constante en ambos ciclos.

La administración de proyectos de software es el arte de balancear los objetivos, los riesgos y las restricciones para entregar un producto que satisfaga las necesidades del cliente y los usuarios finales. Sus propósitos son:

- Proveer de un marco de trabajo para administrar proyectos de software
- Proveer líneas guía para planear, organizar, ejecutar y controlar proyectos
- Proveer un marco de trabajo para administrar el riesgo.

1.9. Abstracción y modelos

La *abstracción* es una actividad mental que consiste en resaltar, disminuir o ignorar la importancia de los hechos u objetos de la realidad para analizar una situación dada. Cuando construimos un modelo resalta el concepto de abstracción, dado que los modelos están orientados hacia la representación de la realidad bajo determinada perspectiva.

Modelos

Los modelos, como resultado de un proceso mental, buscan comunicar una idea-concepto a otra persona. Esto se presenta porque cada persona cuenta con un marco referencial (ideas, estudios, experiencias, creencias) distinto, respecto al de las otras personas; ello hace necesario contar con una herramienta, en este caso, los modelos, para poder comunicar nuestras ideas.

1.10. Modelos de ciclos de vida de los sistemas más importantes

La importancia de los modelos que a continuación se mencionan se sustenta bajo el hecho de que son los más utilizados en *la industria del software*.

CASCADA (Ciclo de Vida clásico, secuencial, lineal)

Descripción	Aplicación	Ventajas	Desventajas
Las fases se aplican de manera secuencial donde no se puede avanzar a la siguiente fase, hasta haber terminado la anterior.	Cuando la complejidad de los sistemas es baja.	Adecuado para sistemas pequeños y de complejidad baja.	▪ Si se comete un error en alguna fase, éste repercute en la fase subsecuente, pero no se descubre sino hasta que se tiene el sistema. ▪ El sistema sólo se ve funcionando hasta el final, por lo que el usuario se desespera al no ver resultados. ▪ Las personas responsables de las fases subsecuentes no pueden empezar hasta que terminen las personas de la fase antecedente.

PROTOTIPOS

Descripción	Aplicación	Ventajas	Desventajas
Se crean prototipos (con o sin funcionamiento) para que el usuario aclare los requerimientos específicos.	Cuando el usuario tiene claro el objetivo o necesidad general, pero no tiene claro los requerimientos específicos.	▪ Existe un acercamiento con el usuario en etapas tempranas del proyecto. ▪ Se puede aplicar o adaptar a otro ciclo de vida de sistemas.	▪ El usuario puede creer que es el sistema final. ▪ No todos los prototipos funcionan. Pueden ser una interfaz gráfica sin comportamiento alguno. ▪ Por lo regular, los prototipos no consideran la calidad del sistema.

INCREMENTAL

Descripción	Aplicación	Ventajas	Desventajas
Su nombre se deriva de que su objetivo es crear el sistema a través de una serie de incrementos. Cada incremento tiene su propio ciclo de vida de sistemas. Cuando se termina un incremento, se integra al sistema. Los incrementos se hacen en paralelo.	Cuando no se cuenta con el personal suficiente para desarrollar todo el sistema.	▪ El usuario puede ver “resultados” desde el primer incremento. ▪ Permite realizar una planeación para maneja los riesgos técnicos.	Para lograr la aplicación de este modelo es necesario contar con todos los requerimientos necesarios para poder planear cada uno de los incrementos.

MODELO EVOLUTIVO (Desarrollo Rápido de Aplicaciones)

Descripción	Aplicación	Ventajas	Desventajas
Es una extensión del modelo incremental, pero los incrementos se hacen de manera secuencial y no en paralelo. Los incrementos que se van desarrollando son aquellos que están claros. Conforme se va avanzando, los requerimientos confusos o ambiguos se van clarificando.	Cuando la complejidad del sistema es alta.	Se entrega la funcionalidad del sistema desde el principio.	▪ No es apropiado cuando los riesgos técnicos son muy grandes. ▪ Se requiere de varias personas para desarrollar los incrementos si se trata de un sistema grande. ▪ Si el sistema no se modula apropiadamente, la delegación del trabajo se vuelve complicada.

MODELO EN ESPIRAL

Descripción	Aplicación	Ventajas	Desventajas
Creado por Bohem en los años 80, integra el modelo de prototipos y el modelo en cascada. Dirige su aplicación mitigando los riesgos técnicos con mayor probabilidad e impacto. Se aplica una serie de ciclos en cascada, pero en espiral, lo que le dio origen a su nombre. Al finalizar cada ciclo, o un poco antes, se realiza la planeación del siguiente ciclo.	Cuando la complejidad del sistema es alta.	▪ Se ajusta el plan de trabajo de cada ciclo, por lo que siempre está actualizado. ▪ Se afrontan los riesgos más peligrosos desde un principio, por lo que si uno no se puede resolver, se minimizan las pérdidas económicas.	Se requiere de personas que puedan detectar los riesgos y generar los planes de contingencia o contención, lo cual implica una gran experiencia y conocimiento por parte de los creadores, al igual que un costo elevado.

1.11. Procesos de desarrollo más importantes

Hay que tomar en cuenta que el ciclo de vida de los sistemas difiere de lo que es un proceso de desarrollo de software, también conocido como modelo de proceso de software. Al respecto, Weitzenfeld (en Álvarez, Weitzenfeld y Cairo, 2002) dice lo siguiente:

> Un modelo de proceso de software define cómo solucionar la problemática del desarrollo de sistemas de software. Para desarrollar software se requiere resolver ciertas fases de su proceso, las cuales se conocen en su conjunto como el ciclo de vida del desarrollo de software. Un modelo de proceso debe considerar una variedad de aspectos, como el conjunto de personas, estructuras organizacionales, reglas, políticas, actividades, componentes de software, metodologías y herramientas utilizadas. (p. 76-77)

Cualquier enfoque de ingeniería debe apoyarse sobre un compromiso de organización de calidad. La *ingeniería de software* es una tecnología multicapa cuyo fundamento es la capa de **proceso**. El proceso de la ingeniería de software es la unión que mantiene juntas las capas de tecnología y permite un desarrollo racional y oportuno de la ingeniería de software. Este proceso define un marco de trabajo para un conjunto de áreas clave de proceso que se deben establecer para la entrega efectiva de la tecnología de la ingeniería de software.

Las áreas clave del proceso forman la base de control de gestión de proyectos del software, se asegura la calidad; se establecen hitos y el contexto en el que se

aplican los métodos técnicos; se obtienen productos del trabajo (modelos, documentos, datos, informes, formularios, etc.) y el cambio se gestiona adecuadamente.

Los **métodos** de la ingeniería del software indican *cómo* construir técnicamente el software y abarcan una gran gama de tareas que incluyen el análisis de requisitos, el diseño, la construcción de programas, las pruebas y el mantenimiento. Estos métodos dependen de un conjunto de principios básicos que gobiernan cada área de la tecnología e incluyen actividades de modelado, así como otras técnicas descriptivas.

A pesar de que existen varios tipos, los procesos de desarrollo de software más utilizados en la industria son:

- *Rational Unified Process*
- *Extreme Programming*;

RUP (*Rational Unified Process* - Proceso Unificado Racional) es un proceso de ingeniería de software desarrollado y mantenido por *Rational Software* que trabaja conjuntamente con UML y es la metodología estándar y más utilizada para análisis, diseño, implementación y documentación de desarrollos orientados a objetos.

Proporciona una estrategia para asignar tareas y responsabilidades en el equipo de desarrollo. Su meta es asegurar la producción de software con calidad, de acuerdo con las necesidades de los usuarios finales (la fecha y el presupuesto planeados). Está integrado por una suite de herramientas de desarrollo.

También es el marco de trabajo de un proceso que puede ser adaptado y extendido, según las necesidades de la organización que lo adopta. Asimismo, captura seis buenas prácticas del desarrollo de software moderno.

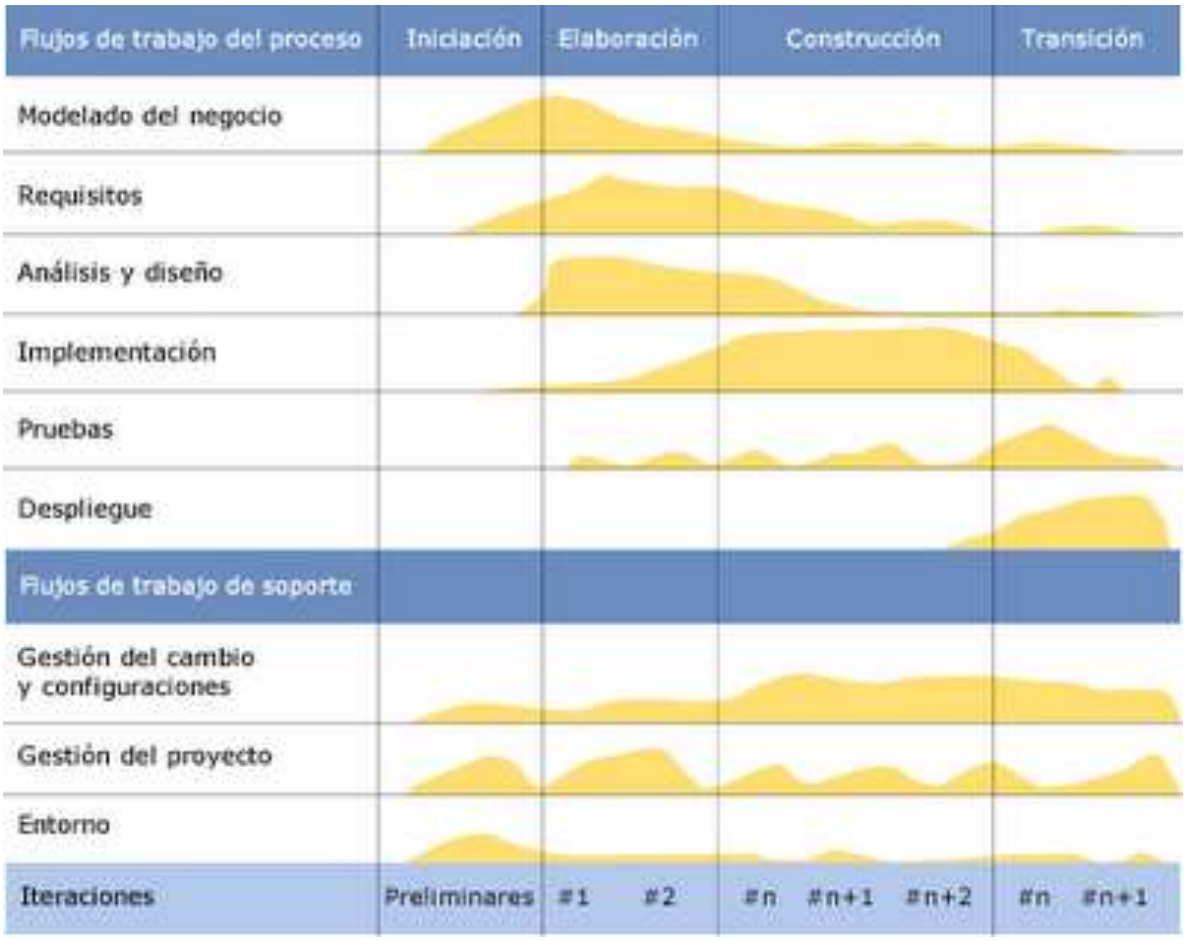
Buena práctica	Descripción
Desarrollo iterativo de software	Uno de los problemas que tiene el modelo de Cascada es la acumulación de errores al aplicarse las etapas en forma secuencial y sin retroalimentación, y sólo saldrán a flote reflejándose en la inconformidad y disgusto de los clientes o usuarios. Para evitar este problema, RUP utiliza un modelo iterativo incremental que se basa en el modelo en Espiral que propone Bohem; este modelo utiliza la estrategia de mitigar los riesgos principales desde las iteraciones iniciales del proyecto. Sus ventajas son las siguientes: evita malentendidos; retroalimentación constante con el usuario; identificación pronta de los requerimientos, diseño e implementación; se ejecutan las pruebas desde la etapas iniciales; se obtienen lecciones aprendidas aplicables en la mejora del proceso, y durante la vida del proyecto se puede saber su estado.
Administración de requerimientos	El desarrollo de un sistema se desenvuelve en un ambiente altamente dinámico, por lo que las necesidades y requerimientos tienen que adecuarse a estos cambios. Por tal motivo, se necesita tener un proceso que identifique estos cambios cuando se presenten y hacer los ajustes necesarios en la planeación. Las ventajas que tenemos al contar con una administración de requerimientos son las siguientes: la comunicación se basa en la definición de requerimientos; los requerimientos se priorizan, se filtran y se rastrean; las inconsistencias son detectadas más fácilmente; con alguna herramienta de apoyo, es posible proveer un repositorio de los requerimientos, atributos y fuente de los requerimientos con enlaces a documentos externos.

Arquitectura basada en el uso de componentes	Para visualizar, especificar, construir y documentar un sistema se necesitan varias perspectivas. La arquitectura de un sistema se utiliza para manejar los diferentes puntos de vista de los *stakeholders*. La arquitectura define la organización, los elementos estructurales y sus interfaces entre ellos y el comportamiento de los elementos. El desarrollo basado en componentes es una estrategia importante para la arquitectura del software porque permite reutilizar y configurar miles de componentes de diferentes fuentes comerciales. Las ventajas de contar con el uso de componentes basados en la arquitectura son los siguientes: facilita la creación de arquitecturas resistentes; permite modularización clara; los componentes proveen una base natural para la administración de la configuración; herramientas de modelado proveen la automatización para el desarrollo de componentes.
Modelado visual del software	Un modelo es una simplificación de la realidad que describe un sistema desde una perspectiva particular. Se construyen modelos para entender mejor el sistema, debido a que no podemos comprenderlo en su totalidad. La actividad de modelado es importante porque ayuda al equipo de desarrollo a visualizar, especificar, construir y documentar la estructura y comportamiento de la arquitectura de un sistema. Las herramientas de modelado visual facilitan la administración de los modelos, permitiendo ocultar o exponer detalles, según sea necesario. El modelado visual también ayuda a mantener la consistencia en la gran cantidad de documentos que se generan en cuanto a los requerimientos, el diseño y la implementación. En breve, el modelado visual ayuda al equipo a administrar la complejidad del software.
Verificación continua de la	Verificar la funcionalidad del sistema involucra crear pruebas para cada escenario principal; cada uno representa algún aspecto del comportamiento esperado del sistema. Si se ocupa un desarrollo

calidad del software	iterativo, se debe probar cada iteración, ya que un proceso continuo asegura la calidad. Con esta forma de trabajo tenemos las siguientes ventajas: se detectan las inconsistencias en los requerimientos, diseño e implementación; las pruebas y verificación están enfocadas en las áreas de alto riesgo; los defectos que son identificados prontamente, y reducen el costo de reparación.
Control de cambios del software	Un desafío clave en el desarrollo de sistemas se afronta con múltiples desarrolladores organizados en equipos, posiblemente en diferentes sitios o trabajando juntos en múltiples iteraciones, versiones, productos y plataformas. En la ausencia de una disciplina de control, el proceso de desarrollo degenera rápidamente en el caos. Coordinar las actividades y los artefactos de los desarrolladores y equipos implica establecer un flujo de trabajo repetible para administrar los cambios al software y otros productos de trabajo de desarrollo. Esta coordinación permite una mejor ubicación de los recursos basada en las prioridades y riesgos del proyecto, y se administra activamente el trabajo de estos cambios a través de las iteraciones. Los cambios de peticiones facilitan una comunicación clara; los espacios de trabajo aislados reducen la interferencia entre los miembros del equipo que trabajan en paralelo.

Buenas Prácticas de RUP (ver, Rational Software, 1998, aquí)

El proceso de RUP está definido en dos dimensiones:



Rational Unified Process (http://commons.wikimedia.org/wiki/File:Rup_espanol.gif?uselang=es, recuperado el 24/03/12)

El eje horizontal

Representa el aspecto dinámico del proceso y está definido en términos de ciclos, fases, iteraciones e hitos y muestra el ciclo de vida del proceso de desarrollo en fases divididas en iteraciones.

El eje vertical

Representa los procesos de las disciplinas principales, las cuales agrupan lógicamente las actividades por su naturaleza. Esta dimensión representa el aspecto estático del proceso y está definido en términos de componentes del proceso, actividades, disciplinas, productos de trabajo y roles.

Iniciación

La meta de esta fase es alcanzar un común acuerdo entre todos los *stakeholders* en cuanto a los objetivos del proyecto, los cuales son:

- Establecer el alcance del proyecto de software y condiciones del ambiente, incluyendo concepción operacional, criterios de aceptación y descripciones de qué es y qué no es parte del producto.
- Discriminar los casos de uso crítico del sistema, es decir, los escenarios primarios del comportamiento que debe manejar la funcionalidad del sistema.
- Exhibir y demostrar, al menos, una arquitectura candidata de algunos de los escenarios primarios.
- Estimar el costo total y el calendario de todo el proyecto para hacer un estimado detallado de la fase de elaboración.
- Identificar los riesgos.

Elaboración

El propósito de la fase de elaboración es analizar el domino del problema y establecer los cimientos de la arquitectura, así como desarrollar el plan del proyecto y eliminar los elementos de alto riesgo presentes. Para complementar estos objetivos se debe tener una vista del sistema “en amplitud y profundidad”.

Las decisiones arquitectónicas se deben hacer con un entendimiento amplio del sistema: de su alcance, su funcionalidad y sus requerimientos no funcionales. Los objetivos de esta etapa son:

- Definir, validar y controlar la arquitectura tan rápido y práctico como sea posible.
- Demostrar que la arquitectura soporta la visión con costo y tiempo razonable.

Construcción

Durante la fase de construcción, todos los componentes y características de la aplicación restantes son desarrollados e integrados en el producto para poder ser probados. En cierto sentido, esta fase establece un proceso de manufactura que se enfoca en la administración de los recursos y en controlar las operaciones para optimizar los costos, los calendarios y la calidad.

Los objetivos son:

- Minimizar los costos de desarrollo para optimizar recursos y evitar el retrabado.
- Alcanzar una calidad adecuada.
- Realizar versiones útiles.

DISCIPLINAS

Modelado Negocio

Sus propósitos son:

- Entender la estructura y la dinámica de la organización con la que se desarrollará el sistema
- Entender el problema correcto en la organización destino e identificar mejoras potenciales
- Asegurar que los clientes, usuarios finales y desarrolladores tengan un común entendimiento en la organización destino
- Identificar los requerimientos del sistema para apoyar la organización destino.

Requerimientos

Describe cómo definir la visión del sistema y traducir la visión a un modelo de casos de uso que, con el apoyo de especificaciones suplementarias, defina los requerimientos detallados de software. Además, describe cómo los atributos de los requerimientos ayudan a administrar el alcance y cambiar los requerimientos en el sistema. Sus metas son:

Establecer y mantener el acuerdo entre los clientes y los *stakeholders* sobre lo que debe hacer el sistema y por qué.
Provee a los desarrolladores un mejor entendimiento de los requerimientos.
Define los límites del sistema.
Provee una base para la planeación de cada iteración.
Provee una base para estimar el costo y el tiempo de desarrollo del sistema.

Análisis y Diseño

Su propósito es traducir los requerimientos a especificaciones que describan cómo implementar el sistema. Para hacer esta traducción se deben entender los requerimientos y transformarlos a un diseño de sistema para seleccionar la mejor estrategia de implementación.

Implementación

Sus propósitos son:

- Definir la organización del código en términos de implementación de subsistemas organizados en capas
- Implementar clases y objetos en términos de componentes (archivos fuente, binarios, ejecutables y otros)
- Probar los componentes desarrollados como unidades
- Integrar en un sistema ejecutable los resultados producidos por implementadores individuales o por equipos.

Pruebas

Esta disciplina es un proveedor de servicios para otras disciplinas. Las pruebas se enfocan principalmente en evaluar y asegurar la calidad del producto usando prácticas como las siguientes:

- Encontrar y documentar la fallas en el producto de software: defectos y problemas
- Aconsejar a la administración con una perspectiva de calidad del software
- Validar que el producto de software trabaje como se diseñó
- Validar que los requerimientos están implementados apropiadamente

Despliegue

Llevar el producto de software al usuario con las siguientes actividades:

Probar el software en un ambiente final de operación
Empaquetar el software para su entrega
Distribuir el software
Instalar el software
Capacitar a los usuarios finales y a los vendedores
Migrar el software existente o convertir bases de datos

Administración de Cambios y Configuración

Su propósito es rastrear y mantener la integridad de la evolución de los activos del proyecto. Durante el desarrollo muchos productos de trabajo son creados, los cuales son activos importantes que deben ser salvaguardados y estar disponibles para volver a usarlos. Estos productos de trabajo evolucionan, especialmente en un desarrollo iterativo, de manera que son actualizados una y otra vez. A menos que una persona sea usualmente responsable de un artefacto, no podemos confiar en la memoria individual para guardar los activos de cada proyecto.

Ambiente

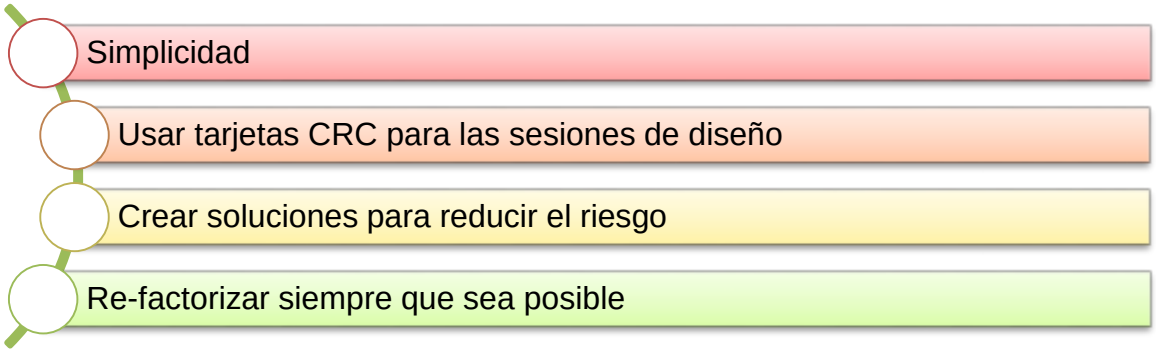
- Apoyar el desarrollo de la organización con procesos y herramientas.
- Seleccionar y adquirir herramientas
- Configurar las herramientas en la organización
- Configurar el proceso
- Mejorar el proceso
- Proporcionar apoyo técnico a los procesos de infraestructura de tecnologías de información, administración de contabilidad y recuperación

XP (*Extreme Programming* - Programación Extrema) tiene como objetivo principal disminuir el costo del cambio en los requerimientos a través de valores, principios, prácticas, etcétera. XP intenta mantener relativamente fijo el costo del cambio en cada una de las etapas del proyecto; asimismo, cuenta con cuatro áreas y cada una define un conjunto de prácticas:

Planeación

- Redacción de historias de usuario
- Creación de calendarios de planeación
- Creación frecuente de versiones pequeñas
- División del proyecto en iteraciones
- La iteración de planeación inicia cada iteración
- Movimiento alrededor de la gente

Diseño



Codificación

- El cliente siempre está disponible
- El código debe ser escrito de acuerdo con los estándares
- Codificar la primera prueba de unidad
- Todo código es programado por pares
- Únicamente un par integra el código a la vez
- Usar código colaborativo
- Dejar la optimización hasta el último

Pruebas

- Todo código debe tener pruebas unitarias
- Todo código debe pasar todas las unidades antes de que pueda ponerse en marcha
- Cuando se encuentra un bug, deben crearse las pruebas
- Las pruebas de aceptación se ejecutan y las estadísticas se publican

1.12. Enfoques del desarrollo de sistemas

1.12.1. Enfoque Estructurado

Su concepto principal es el de función. Bajo esta perspectiva, el *enfoque estructurado* analiza y diseña los sistemas con un enfoque de descomposición funcional.

1.12.2. Enfoque Orientado a Objetos

Su concepto principal es el de objeto. Bajo esta perspectiva, el *enfoque orientado objetos* analiza y diseña los sistemas con un enfoque de responsabilidades a través de objetos que están presentes en la situación real, ya sea de carácter físico o abstracto.

RESUMEN

Los sistemas y la información son los elementos básicos que conforman a los llamados “sistemas de información”.

Para desarrollar un sistema de información se hace uso de las etapas que conforman el ciclo de vida de los sistemas: análisis, diseño, implementación, pruebas, implantación, mantenimiento.

Al entender el concepto de sistema podemos analizar las situaciones con una perspectiva integradora donde cada elemento o componente que interviene modifica el comportamiento del todo, es decir, del sistema.

BIBLIOGRAFIA

SUGERIDA

Autor	Capítulo	Páginas
Bird (2000)	2 y 17	30-53 y 652
Ceballos (2002)	1 y 10	5, 6 y 342
Cairó (2002a)	13 y 4	231 y 273
Cairó (2002b)	1	36

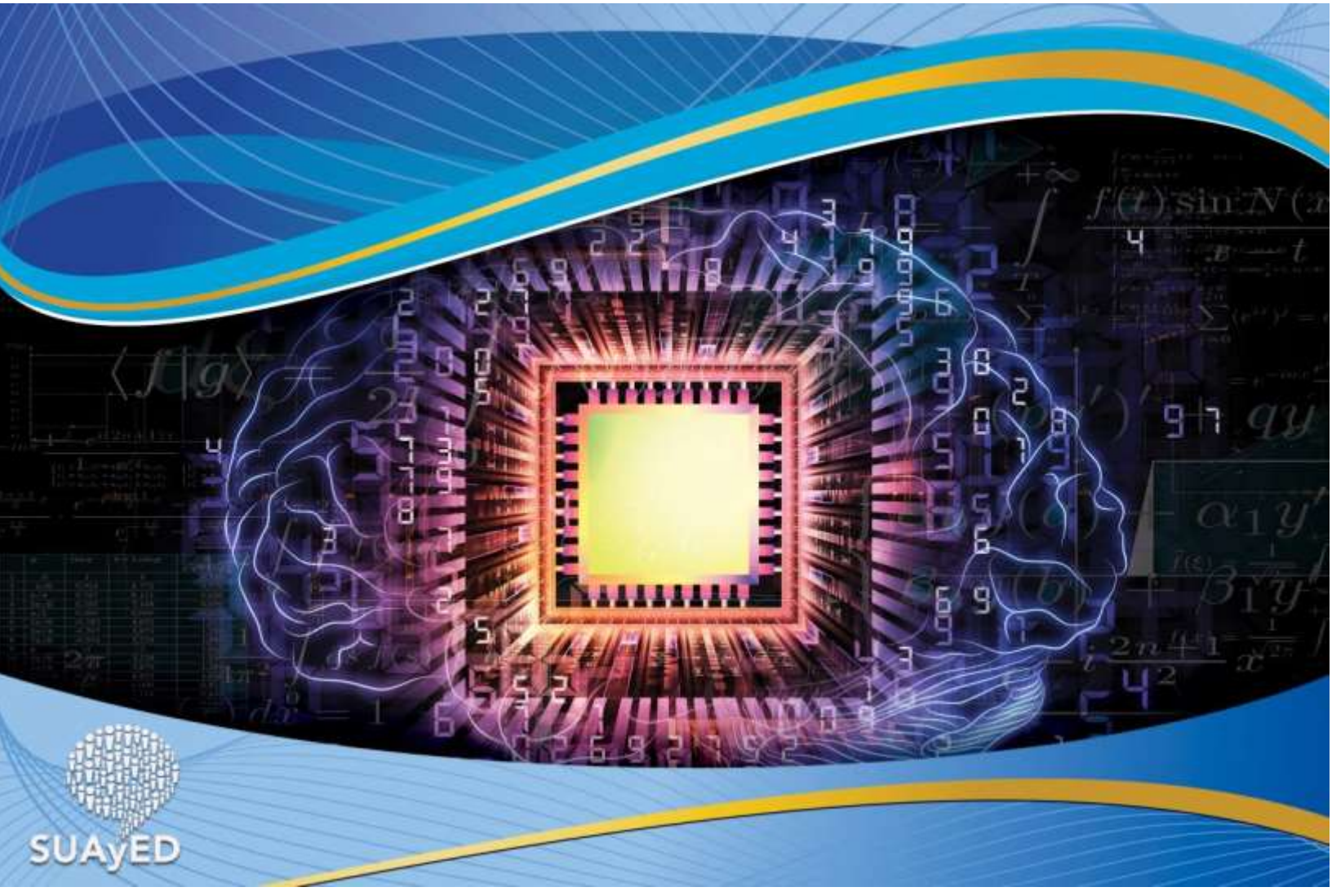
Bird, Richard. (2000). *Introducción a la programación funcional con Haskel*. (2ª ed.) México: Prentice Hall

Ceballos, Francisco Javier. (2002). *Enciclopedia del lenguaje C.* México: Alfaomega / Ra-Ma.

Cairó Batistutti, Oswaldo. (2002a). *Metodología de la programación. Algoritmos, Diagramas de flujo y programas*, México: Alfaomega

Unidad 2

Análisis de sistemas



OBJETIVO PARTICULAR

Al término de la unidad, el alumno podrá identificar las actividades y técnicas más relevantes de la administración de requerimientos y del análisis de sistemas que son utilizadas en el enfoque estructurado.

TEMARIO DETALLADO

2. Análisis de Sistemas

- 2.1. Principios del Análisis Estructurado
- 2.2. Participantes en el análisis estructurado
- 2.3. Perfil del analista de sistemas
  - 2.3.1. Responsabilidades
  - 2.3.2. Conocimientos
  - 2.3.3. Habilidades
- 2.4. Administración de Requerimientos
  - 2.4.1. Definición de requerimientos
  - 2.4.2. Identificación de requerimientos
  - 2.4.3. Técnicas de Recopilación de Información
    - 2.4.3.1. Entrevistas
    - 2.4.3.2. Cuestionarios
    - 2.4.3.3. Revisión de Registros, Reportes, Formularios
    - 2.4.3.4. Observación
    - 2.4.3.5. Método Delphi

2.4.3.6. Desarrollo Conjunto de Aplicaciones

2.4.4. Análisis y Clasificación de los requerimientos

2.4.5. Negociación de los requerimientos

2.4.6. Especificación de los requerimientos

2.4.7. Técnicas para la Especificación de Requerimientos

2.4.7.1. Árboles y Tablas de Decisión

2.4.7.2. Español Estructurado

2.4.7.3. Diagramas de Flujo

2.4.7.4. Especificación de Casos de Usos

2.5. Modelado del Análisis

2.5.1. Modelado Ambiental

2.5.1.1. Diagramas de Contexto (DC)

2.5.2. Modelado de Datos

2.5.2.1. Diagrama Entidad-Relación (DER)

2.5.2.2. Diccionario de datos

2.5.3. Modelado de Comportamiento

2.5.3.1. Diagrama de Flujo de Datos (DFD)

2.5.3.2. Diagrama de Transición de Estados (DTE)

2.5.3.3. Diccionario de datos

2.6. Estudio de Factibilidad y Análisis Costo-Beneficio

INTRODUCCIÓN

En todo análisis buscamos representar la realidad de la organización desde la perspectiva dinámica y estática, para lograrlo hacemos uso de diferentes diagramas. En el ciclo de vida de los sistemas la primera etapa es la del **análisis**. Esta etapa tiene como propósito identificar cuál es la problemática por resolver y cuál es el comportamiento del sistema actual, ya sea que esté o no automatizado; no es suficiente conocer el problema, las causas deben ser detectadas para poder atacarlas y dar una solución adecuada. Asimismo, se tiene como objetivo identificar las necesidades y los requerimientos de los participantes que deben ser contemplados para construir el sistema.

Existen diferentes metodologías para ejecutar el Análisis Estructurado, al igual que notaciones y simbologías. Los autores más representativos del tema son:

	Método de Gane y Sarsons	Método de DeMarco	Método de Jourdon
DIFERENCIAS	− Construir un modelo lógico actual. − Construir un modelo lógico del nuevo sistema. − Seleccionar un modelo lógico. − Crear un nuevo modelo físico del sistema. − Empaquetar la especificación.	− Construir un modelo físico actual. − Crear un conjunto de modelos alternativos. − Examinar los costos y tiempos de cada opción. − Empaquetar la especificación.	− Realizar los diagramas de flujo del sistema. − Realizar el diagrama de estructuras evaluar el diseño, midiendo la calidad cohesión y el acoplamiento. − Preparar el diseño para la implantación.

Tabla 2.1: Diferencias entre las metodologías convencionales de Gane y Sarsons, DeMarco y Jourdon, en (Valiente, 2008)

2.1. Principios del análisis estructurado

El diseño de la arquitectura del software es como el plano para construir una casa porque describe la estructura y organización de los componentes del software, sus propiedades y conexiones. Cuando se diseña un sistema informático se está creando la comunicación entre el hombre y la máquina, de allí que se deba tener en cuenta los requisitos del usuario, de la tarea y del entorno bajo los siguientes principios:

- Enfocar el modelo de análisis solamente a los requisitos que ayuden a resolver el problema o satisfacer la necesidad.
- Posponer la toma de decisiones que conciernen a la infraestructura hasta la etapa de diseño.
- Tratar de minimizar el acoplamiento de los módulos del sistema.
- Asegurar que la documentación que se genere aporte valor a la mayoría de los interesados.
- Tratar de mantener el análisis lo más simple posible.

Los principios anteriormente mencionados no son absolutos sino que pueden ser adaptados de acuerdo con las características de la organización o del sistema.

2.2. Participantes en el análisis estructurado

Antes participaban sólo analistas y se creía innecesario incluir a diseñadores, programadores, administradores de base de datos o cualquier otro rol que interviene directamente en las etapas subsecuentes del análisis. Con el tiempo esto ha cambiado: Sus conocimientos y experiencias aportan elementos valiosos para determinar si se tiene o no la factibilidad técnica, operativa y económica para desarrollar el sistema.

Después fueron los usuarios los que quedaban relegados porque a menudo es difícil lograr que los usuarios colaboren en un proyecto de desarrollo, sobre todo si están escasos de tiempo o conocimientos. Sin embargo, incluso si la participación de los usuarios es mínima, la experiencia práctica con el sistema ayuda a los usuarios a apreciar sus beneficios y da pie a útiles sugerencias para mejorarlo (De y Ferratt, 1998) con el fin de que todos los participantes tengan el mismo concepto del sistema.

Además, la relación entre los especialistas en sistemas de información y los clientes o usuarios finales ha sido tradicionalmente complicada para terminar un proyecto. Los especialistas a menudo dan soluciones técnicas sofisticadas para la resolución de problemas, en las que se optimiza la eficiencia del hardware y software, a expensas de la facilidad de uso o de la eficiencia de la organización.

Actualmente, un analista de sistemas que busca un análisis eficaz al delimitar el alcance del sistema, identificar el problema, sus causas y requerimientos, debe apoyarse de la participación activa de:

2.3. Perfil del analista de sistemas

El analista de sistemas es una persona experta en la identificación y comprensión de los problemas y oportunidades. Es una persona con madurez, visión y una amplia experiencia para conjugar sus conocimientos y juicios al momento de crear soluciones y actúan como enlace entre la ingeniería de sistemas y los usuarios.

Según Senn (1992, p. 12), dependiendo de las funciones de un analista de sistemas se puede clasificar en: Analista de sistemas, Analista y diseñador de sistemas y analista diseñador y programador de sistemas, en donde cada uno se puede identificar y diferenciar de los demás por las actividades que definen sus denominaciones. El analista de sistemas más valioso y mejor calificado es aquel que sabe programar pero no es su única función.

Kendall (2005, p.6) define el perfil de los analistas de sistema basándose en las fases impuestas en el paradigma Ciclo de Vida de Desarrollo de Sistemas (CVDS) como:

Consultor. Cuando se comienza el CVDS el analista cumple en papel de consultor, asesorando a la empresa sobre los mejores métodos y sistemas que se pueden emplear para la óptima gestión de información, recomendando sistemas ya sean de tipo manual o de tipo informático, se puede traducir en una ventaja porque los consultores externos tienen una perspectiva fresca de la cual carecen los demás miembros de una organización. También se puede traducir en una desventaja porque alguien externo nunca conocerá la verdadera cultura organizacional.

Experto de soporte	• En este rol el analista recurre a su experiencia profesional con el hardware y software de cómputo y al uso que se le da en el negocio. Con frecuencia, este trabajo no implica un proyecto completo de sistemas, sino más bien la realización de pequeñas modificaciones o la toma de decisiones que se circunscriben a un solo departamento.

El experto en soporte se identifica con los últimos pasos del CVDS donde el analista se desempeña en el asesoramiento de hardware y software, basado en el conocimiento y especialmente en la experiencia. Sirviendo el analista muchas veces de escalón para hacer que el sistema desarrollado (no liderado por él) tenga éxito.

Agente de cambio	• El rol más completo y de mayor responsabilidad que asume el analista de sistemas es el de agente de cambio, ya sea interno o externo para la empresa. Un agente de cambio se puede definir como alguien que sirve de catalizador para el cambio, desarrolla un plan para el cambio y coopera con los demás para facilitar el cambio.

En su calidad de analista de sistemas desempeñando la función de agente de cambio debe promover un cambio que involucre el uso de los sistemas de información. También es parte de su tarea enseñar a los usuarios el proceso del cambio, ya que las modificaciones a un sistema de información no sólo afectan a éste sino que provocan cambios en el resto de la organización.

2.3.1. Responsabilidades

- Asociar las necesidades con el problema principal por resolver o la oportunidad por conseguir
- Planear las actividades de análisis de sistemas.
- Escoger (o diseñar) y utilizar los métodos, técnicas y herramientas más adecuadas para el análisis del sistema.
- Estudiar el sistema de planeación, organización, dirección y control de la empresa.
- Representar con algún tipo de especificación los procesos que se realizan en la organización y que están relacionados con el sistema por desarrollar.
- Modelar los aspectos dinámicos y estáticos del sistema actual.
- Proponer y aplicar las medidas de carácter organizativo que se requieran para perfeccionar los procesos.
- Revisar los resultados obtenidos y elaborados por los programas.

- Elaborar los datos de prueba para comprobar la calidad de los programas individualmente y en su conjunto.
- Capacitar a los usuarios en la operación del sistema

2.3.2. Conocimientos

- En diversas áreas de cómputo (Bases de datos, redes, programación, sistemas de información, sistemas expertos, etc.)
- Tiene conocimiento sobre negocios y tecnología.
- Tiene experiencia en el dominio del problema.
- Conocimientos generales de la empresa.
- Domina la ingeniería del software.
- Cuenta con certificaciones.
- Asociado con organizaciones de tecnologías de información.

2.3.3. Habilidades

- Es un facilitador con gran capacidad de comunicación
- Es un líder a la hora de dar órdenes, de dirigir los esfuerzos de diferentes equipos y de tomar decisiones críticas bajo presión.
- Es un comunicador: sabe escuchar, persuadir, motivar y guiar.
- Comprende los requerimientos
- Capacidad de abstracción
- Capacidad de síntesis
- Disciplina
- Organización
- Ser meticuloso, incluso con los pequeños detalles cuando la situación lo amerite

- Comprensión de la cultura de organización y su impacto sobre los sistemas de información
- Capacidad de identificar problemas y sus causas
- Administración de los requerimientos
- Modelado del aspecto dinámico y estático del sistema actual

2.4. Administración de los requerimientos

La *administración de requerimientos* comprende las actividades relacionadas con la identificación, especificación, clasificación, seguimiento y control de los requerimientos durante todo el ciclo de vida del desarrollo del sistema. Es una metodología indispensable para el aseguramiento de la calidad de los productos, así como para el control y seguimiento de los proyectos.

2.4.1. Definición de Requerimientos

Un *requerimiento* es la capacidad del software[2].

Estas capacidades del software, también conocidas como “características”, pueden ser solicitadas por los usuarios o ser detectadas por los analistas para satisfacer un contrato, una especificación, un estándar u otra documentación impuesta.

[2] En este caso ‘software’ tiene la misma connotación que sistema: que solicita el usuario, de manera implícita o explícita, para solucionar un problema o para alcanzar un objetivo.

2.4.2. Identificación de los Requerimientos

La identificación de los requerimientos no es una tarea fácil, de hecho, es una de las actividades más complejas durante el análisis. En esta actividad intervienen directamente la percepción del analista, al igual que su conocimiento sobre la empresa o negocio donde se vaya a implantar el sistema, y los intereses organizacionales o particulares del cliente o los usuarios. Además, el vocabulario es otro factor que aumenta la complejidad, debido a que si no se toma en cuenta el contexto, se podrían estar malinterpretando las ideas de los participantes.

Los requerimientos no sólo son proporcionados por los usuarios sino que también provienen de reglamentos, manuales, la competencia, el gobierno, los proveedores, los clientes o cualquier otra fuente que delimite el comportamiento que deba tener el sistema.


Prosumer
Customer
Citizen
Consumer
Producer
Community
Participant

2.4.3. Técnicas de Recopilación de Información

Las técnicas más utilizadas para identificar los requerimientos son la *entrevista* y el *cuestionario*, aunque también se utilizan con menos frecuencia la *revisión de documentación*, como manuales, reportes, estadísticas, reglamentos, políticas, estudios de necesidades, estudios de viabilidad (económica, técnica, operativa y/o legal).

2.4.3.1. Entrevistas

La *entrevista* es un conjunto de preguntas generalmente abiertas (no hay una respuesta exacta) dirigidas a una persona o varias personas que conforman un equipo para obtener datos de carácter cualitativo de manera que los datos no se podrán tabular o manipular estadísticamente. Se busca conocer el punto de vista de los entrevistados según su realidad.

El empleo de las entrevistas en la recopilación de requerimientos se enfoca en buscar los motivos que originan la creación del sistema, las causas que originan el problema o la especificación de los procesos.

Según Kendall (2005, p. 90) los pasos para la planeación de la entrevista son:

1. **Leer los antecedentes**. Leer y entender tanto como sea posible los antecedentes de los entrevistados y su organización.
2. **Establecer los objetivos de la entrevista**. Debe haber de cuatro a seis áreas clave referentes al procesamiento de la información y el comportamiento

relacionado con la toma de decisiones acerca de las cuales tendrá usted que hacer preguntas.

3. **Decidir a quién entrevistar**. Incluir a gente clave de todos los niveles que vayan a ser afectadas por el sistema de alguna manera. Esfuércese por conseguir el equilibrio de tal manera que atienda las necesidades de tantos usuarios como sea posible.
4. **Preparar al entrevistado**. Preparar a la persona que va a ser entrevistada hablándole por anticipado o enviándole un mensaje de correo electrónico y dándole tiempo para pensar en la entrevista. Si va a realizar una entrevista a profundidad, puede enviar sus preguntas por correo electrónico con antelación para darle tiempo al entrevistado a que piense sus respuestas.
5. **Decidir el tipo de preguntas y la estructura**. Escribir preguntas que abarquen las áreas clave de la toma de decisiones que haya descubierto al determinar los objetivos de la entrevista. Las técnicas apropiadas para preguntar son el corazón de la entrevista. Las preguntas tienen algunas formas básicas que usted debe conocer.

2.4.3.2. Cuestionarios

Al igual que la entrevista, es un conjunto de preguntas, pero dirigidas a una cantidad considerable de personas. Las preguntas son generalmente cerradas: verdadero o falso o de opción múltiple.

Los cuestionarios podrían parecer una manera rápida de recopilar grandes cantidades de datos sobre la opinión que los usuarios tienen del sistema actual, sobre los problemas que experimentan con su trabajo y sobre lo que la gente espera de un sistema nuevo o uno modificado pero el

desarrollo de un cuestionario útil implica una considerable cantidad de tiempo de planeación.

A continuación mencionamos algunas directrices servir para decidir cuándo es apropiado el uso de cuestionarios:

1. Las personas que se van a encuestar se encuentran en ubicaciones dispersas (diferentes instalaciones de la misma corporación).

2. Una gran cantidad de personas está involucrada en el proyecto de sistemas, y es importante saber qué proporción de un grupo dado (por ejemplo, los directivos) aprueba o desaprueba una característica específica del sistema propuesto.

3. Al realizar un estudio preliminar, se busca medir la opinión general antes de que se determine el rumbo que tomará el proyecto de sistemas.

4. Se desea tener la certeza de que en las entrevistas de seguimiento se identificará y abordará cualquier problema relacionado con el sistema actual.

2.4.3.3. Revisión de registros, reportes, formularios y otros

En las organizaciones existe una gran cantidad de documentación que debe ser analizada: manuales, reportes, oficios, memorándums, etcétera.

Estos documentos son de gran utilidad porque describen, en menor o mayor sentido, los procesos que se llevan a cabo; además, los datos que contienen nos permiten modelar los aspectos de comportamiento y de datos. Si se ve desde el punto de vista del cliente, el proceso de revisar este tipo de documentación agiliza el proceso de desarrollo del sistema porque el analista se presenta con los usuarios y clientes con una idea más clara de lo que hace y cómo se hace en la organización.

4.2.3.4. Observación

Esta es una técnica que siempre utilizan los analistas de sistemas. Se debe prestar atención a las relaciones interpersonales porque nos ayudan a identificar la forma de trabajo en el negocio o empresa, así como para identificar la organización informal existente.

Está técnica puede ser un tanto incómoda para las personas que están siendo observadas, por lo que el analista debe tener tacto a la hora de presentarse en el lugar de trabajo de las personas.

2.4.3.5. Método Delphi

El método Delphi consiste en reunir a un grupo de expertos para solucionar determinados problemas, fue puesto a punto por Olaf Helmer y T.J. Gordon de la Rand Co. a principios de los años 60. Se trataba de interrogar individualmente a una selección de personas, con el objeto de evaluar, vía estimación subjetiva, probabilidades o fechas de acaecimiento de sucesos.

Las características del método Delphi son las siguientes:

1) Anonimato en las respuestas. La opinión de los miembros del grupo se obtiene por medio de un cuestionario formal, siendo consultados a distancia y garantizándoles el anonimato.
2) Iteración controlada. El método consiste en una secuencia de procesos iterativos en los que el resultado de cada uno de ellos sirve de información al siguiente.
3) Respuesta estadística del grupo. La opinión del grupo se define como una agregación conveniente de las opiniones individuales. Calculándose una medida de posición central (la mediana) y otra de dispersión (el espacio intercuartílico).

Al igual que en otras técnicas de evaluación subjetiva, en el método Delphi actúan dos grupos: el grupo de expertos, que son las personas consultadas a distancia y el grupo de analistas, que analiza las respuestas y realiza la evaluación buscada.

El grupo de expertos tiene por misión la elaboración de estimaciones, mediante un proceso de razonamiento, que posteriormente será contrastado, corregido y completado en fases sucesivas de estimación por medio de un encuestamiento activo.
Formarán el grupo de expertos personas que posean un conocimiento de la estructura del sistema en estudio, conocimiento que implicará una capacidad de anticipación al cambio, en la evolución del mismo.

Por otra parte, los analistas tienen por misión controlar el proceso de confirmación del juicio de los expertos, sin introducir en el mismo ningún sesgo.

Sus funciones son:

- Estructuración de los cuestionarios.
- Formulación de las preguntas, con visita personal al encuestado a partir de la segunda iteración.
- Recensión de las respuestas.
- Análisis de las respuestas en sus diferentes etapas: analítica, explicativa y sintética o agregativa.
- Redacción del informe final, con la presentación de datos, esquemas analíticos y sintéticos e interpretación. Modelado de sistemas, 2011)

2.4.3.6. Desarrollo Conjunto de Aplicaciones

Aunque el *Joint Application Development* (JAD) se puede sustituir por las entrevistas personales en cualquier momento apropiado del ciclo de vida del desarrollo de sistemas, normalmente ha sido utilizado como una técnica que le permite, como analista de sistemas, realizar el análisis de los requerimientos y diseñar la interfaz de usuario en conjunto con los usuarios Kendall (2005, p.97).

El desarrollo de aplicaciones JAD es una alternativa a las entrevistas que se desarrolla en grupo en la que participan analistas, usuarios, administradores del sistema y clientes.

Está basada en cuatro principios fundamentales:

Dinámica de grupo
El uso de ayudas visuales para mejorar la comunicación
Mantener un proceso organizado y racional
Una filosofía de documentación WYSIWYG (*What You See Is What You Get*, lo que ve es lo que obtiene).

El equipo de trabajo se reúne en varias sesiones para establecer los requisitos de alto nivel a trabajar, el ámbito del problema y la documentación, llegándose a una serie de conclusiones que se documentan para ir concretando más las necesidades del sistema.

Esta técnica presenta una serie de ventajas frente a las entrevistas tradicionales, ya que ahorra tiempo al evitar que las opiniones de los clientes se tengan que contrastar por separado, pero requiere un grupo de participantes bien integrados y organizados (Cf. Escalona y Koch, 2002, p. 7).

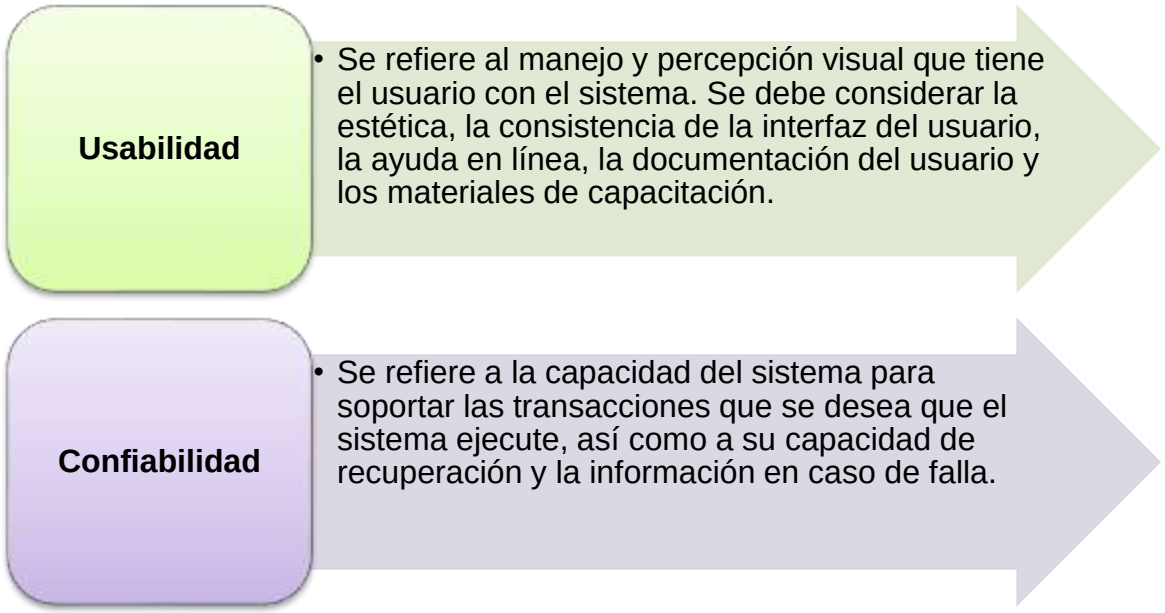
2.4.4. Análisis y clasificación de los Requerimientos

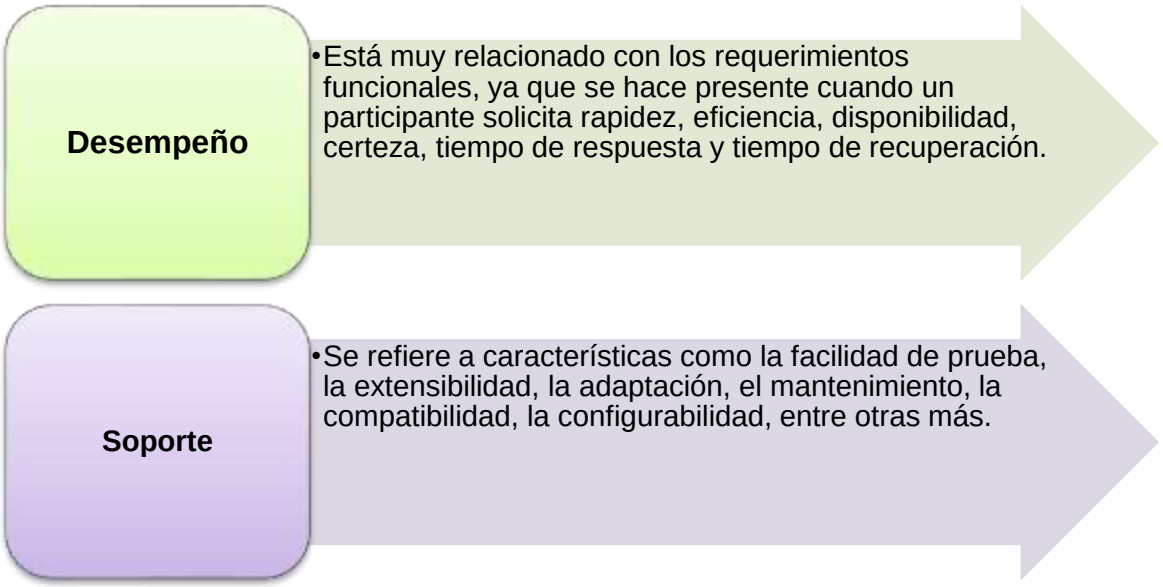
Conforme se desarrolla el sistema se irán detectando algunos requerimientos, por lo que deberán ser analizados, es decir, habrá que clasificarlos, evaluarlos y refinarlos.

Básicamente, los requerimientos se clasifican en *funcionales* y *no funcionales*. Son acciones que el sistema debe desarrollar sin tomar en cuenta las restricciones físicas. Este tipo de requerimientos se describen en tablas de decisión, español estructurado y diagramas de flujo, pero sobre todo con casos de uso. Los requerimientos funcionales especifican el comportamiento interno y externo del

sistema. A su vez, los funcionales se dividen en: Usabilidad, Confiabilidad, Desempeño, Soporte y otros.

Usabilidad: se refiere al manejo y percepción visual que tiene el usuario con el sistema. Se debe considerar la estética, la consistencia de la interfaz del usuario, la ayuda en línea, la documentación del usuario y los materiales de capacitación.
Confiabilidad: se refiere a la capacidad del sistema para soportar las transacciones que se desea que el sistema ejecute, así como a su capacidad de recuperación y la información en caso de falla.
Desempeño: está muy relacionado con los requerimientos funcionales, ya que se hace presente cuando un participante solicita rapidez, eficiencia, disponibilidad, certeza, tiempo de respuesta y tiempo de recuperación.
Soporte: se refiere a características como la facilidad de prueba, la extensibilidad, la adaptación, el mantenimiento, la compatibilidad, la configurabilidad, entre otras más.





Los requerimientos no funcionales describen únicamente atributos del sistema o atributos del ambiente del sistema que no se relacionan directamente con el comportamiento interno o externo del sistema.

2.4.5. Negociación de los Requerimientos

La constante de todo proyecto es la escasez de recursos, motivo por el cual no se podrán atender todas las peticiones de los participantes. Es tarea del analista entender la misión y objetivos de la organización con el objeto de identificar aquellos requerimientos del sistema que serán de ayuda para lograrlos.

Por tal motivo, el momento más indicado para negociar los requerimientos que se desarrollarán es en las primeras reuniones. Las personas que deben asistir a esta reunión son los miembros de equipo de desarrollo, los usuarios y los clientes. El equipo de desarrollo debe dar una estimación del tiempo requerido para cada

requerimiento, los usuarios deben definir cuáles son los requerimientos más importantes y el cliente confirmar si tiene los recursos necesarios para los requerimientos. La habilidad del analista para cruzar la factibilidad, prioridad y costos es determinante, ya que se espera que él pueda dar alternativas que sean de la satisfacción del cliente y que se acoplen a la misión de la organización.

2.4.6. Especificación de los requerimientos

Un error común en el análisis de sistemas es que con una declaración inicial de los requerimientos es más que suficiente para comprender el problema que se busca resolver. Por un lado tenemos que el analista debe seguir un proceso en donde necesita entender el qué, para qué y para quién de la organización; este proceso consume más tiempo si es la primera vez que el analista va a trabajar en el giro al que se dedica la organización. Por el otro lado, los usuarios tienden a pensar cosas que no expresan y dan por hecho que el analista las conoce y entiende.

Estos dos factores enfatizan el cuidado que debe tener al analista al momento de registrar los requerimientos: no puede dar por hecho algo y debe estar consciente de que, conforme se avance en el proyecto, deberá especificar con mayor detalle aquellos requerimientos con una alta complejidad o que sean susceptibles a constantes cambios. Independientemente de la técnica que se haya elegido para registrar los requerimientos, el analista deberá actualizar el registro conforme surjan nuevos hallazgos o cambios en los requerimientos.

Casi siempre, los usuarios o clientes sólo expresarán sus requerimientos de carácter funcional, a pesar de que también existen los requerimientos no funcionales. Estos requerimientos no son sencillos de identificar, sobre todo cuando

se empieza. Una ayuda que proporciona la ingeniería de software para identificar este tipo de requerimientos son las llamadas “cualidades de software”, las cuales se enlistan a continuación (recuperado de FIG, 2011):

Cualidad	**Descripción**
Correcto	Un sistema es correcto si se comporta de acuerdo a la especificación de los requerimientos funcionales que debería proveer. Esta definición de correcto no toma en consideración el que la especificación en sí misma pueda ser incorrecta por contener inconsistencias internas o por no corresponder de forma adecuada a las necesidades para las que fue concebido el programa.
Confiabilidad	La confiabilidad se define en términos del comportamiento estadístico: la probabilidad de que el sistema opere como se espera en un intervalo de tiempo específico. Contrariamente a la cualidad de correcto, que es una cualidad absoluta, la confiabilidad es relativa. Cualquier desviación de los requerimientos hace que el sistema sea incorrecto; y si la consecuencia de un error en el sistema no es seria, el sistema incorrecto aún puede ser confiable.
Robustez	Un sistema es robusto si se comporta de forma razonable, aun en circunstancias que no fueron anticipadas en la especificación de requerimientos; por ejemplo, cuando encuentra datos de entrada incorrectos o algún malfuncionamiento del hardware. Un sistema que genere un error no recuperable en tiempo de ejecución, tan pronto

	como el usuario ingrese inadvertidamente una instrucción incorrecta, no será robusto; de cualquier manera, podría ser correcto si en la especificación de requerimientos no se establece la acción por tomar si se ingresa una instrucción incorrecta.
Eficiencia o Desempeño	Un sistema es eficiente si utiliza los recursos en forma económica. El desempeño de un sistema es importante porque afecta su usabilidad. Por ejemplo: si es muy lento, reduce la productividad de los usuarios; si usa demasiado espacio de disco, puede ser muy caro de ejecutar; si utiliza demasiada memoria, puede afectar al resto de las aplicaciones que se están ejecutando. La visión de “demasiado caro” cambia constantemente según los avances tecnológicos: actualmente, una computadora es más barata que hace unos años y son bastante más poderosas.
Amigabilidad	Un sistema es amigable si un usuario humano lo encuentra fácil de utilizar. Esta definición refleja la naturaleza subjetiva de la amigabilidad: un sistema utilizado por usuarios no experimentado califica como amigable por varias propiedades distintas a las de un sistema utilizado por usuarios expertos.
Verificabilidad	Un sistema es verificable si sus propiedades pueden ser verificadas fácilmente. Por ejemplo: la cualidad de correcto o el desempeño de un sistema son propiedades que interesan verificar. El diseño modular, las prácticas de codificación disciplinadas y la utilización de lenguajes de programación adecuados contribuyen a la verificabilidad

	de un sistema. La verificabilidad es, en general, una cualidad interna que a veces puede ser externa.
Mantenibilidad	El término “mantenimiento” es utilizado generalmente para referirse a las modificaciones que se realizan a un sistema, luego de su liberación inicial, siendo visto simplemente como “corrección de bugs”. Algunos estudios han mostrado que la mayor parte del tiempo utilizado en el mantenimiento es para agregarle al producto características que no estaban en las especificaciones originales. En realidad la palabra “mantenimiento” no es apropiada para el software, ya que cubre un amplio rango de actividades que tienen que ver con la modificación de un software existente para lograr una mejora. Un término óptimo para este proceso es “evolución del software”. Para analizar los factores que afectan estos costos, es usual dividir el mantenimiento del software en tres categorías: de corrección, de adaptación y de mejora.
Reparabilidad	Un sistema es reparable si permite la corrección de sus defectos con una carga limitada de trabajo. En el software las partes no se deterioran, y aunque el uso de partes estándares puede reducir el costo de producción del sistema, el concepto de partes reemplazables pareciera no aplicar a la reparabilidad del sistema. Otra diferencia es que el costo del software está determinado no por partes tangibles sino por actividades de diseño. Un producto de software consistente en módulos bien diseñados es más fácil de analizar y reparar que uno

	monolítico; sin embargo, el solo aumento del número de módulos no hace que un producto sea más reparable. Una modularización adecuada, con definición adecuada de interfaces que reduzcan la necesidad de conexiones entre los módulos, promueve la reparabilidad debido a que permite que los errores estén ubicados en unos pocos módulos, facilitando la localización y eliminación de los mismos.
Evolucionabilidad	Un sistema es evolucionable si acepta cambios que le permitan satisfacer nuevos requerimientos. En el caso de sistemas, en general, la implementación del cambio se comienza sin realizar algún estudio de factibilidad, dejando únicamente el diseño original, sin actualizar las especificaciones para reflejarlo; ello hace que cambios futuros sean cada vez más difíciles de aplicar. La evolucionabilidad es una cualidad tanto del producto como del proceso, la cual debe ser capaz de adaptarse a nuevas técnicas de gestión y organización, así como a cambios en la educación en ingeniería, etc. Es una de las cualidades más importantes del software e involucra otros conceptos, como “familias de programas”.
Reusabilidad	La reusabilidad es similar a la evolucionabilidad: en la segunda se modifica un producto para construir una nueva versión del mismo producto, en la primera se utiliza un producto, posiblemente con modificaciones menores, para construir otro producto. Es difícil lograr la reusabilidad terminado el sistema, sin embargo, eso se debe contemplar al momento de

	desarrollar los componentes del software. Otro nivel de reusabilidad está dado en los requerimientos: al analizar una nueva aplicación se pueden identificar partes que son similares a otras, utilizadas en una aplicación previa; también se nota en el nivel del código, del cual se pueden reutilizar componentes desarrollados en una aplicación anterior.
Portabilidad	El sistema es portable si puede ser ejecutado en distintos ambientes, refiriéndose este último tanto a las plataformas de hardware como a los ambientes de software en determinado sistema operativo. Si bien se ha transformado en un tema importante debido a la proliferación de procesadores y sistemas operativos distintos, puede ser importante incluso en una misma familia de procesadores debido a las variaciones de capacidad de memoria e instrucciones adicionales, por lo que una forma de lograr portabilidad es asumir una configuración mínima y utilizar un subconjunto de las facilidades provistas que se garantiza estarán disponibles en todos los modelos de la arquitectura, como instrucciones de máquina y facilidades del sistema operativo. También es necesario utilizar técnicas que permitan al software determinar las capacidades del hardware y adaptarse a éstas. En general, la portabilidad se refiere a la habilidad de un sistema de ser ejecutado en plataformas de hardware distintas y, a medida que la razón de dinero gastado en software contra hardware crece, la portabilidad gana importancia.

Comprensibilidad	La comprensibilidad es una cualidad interna del producto que ayuda a lograr muchas de las otras cualidades, como la evolucionabilidad y la verificabilidad. Desde un punto de vista externo, un usuario considera que un sistema es comprensible si su comportamiento es predecible; en este caso, la comprensibilidad es un componente de la amigabilidad con el usuario.
Interoperabilidad	La interoperabilidad se refiere a la habilidad de un sistema de coexistir y cooperar con otros sistemas. Un concepto relacionado con la interoperabilidad es el de “sistema abierto”, el cual es una colección extensible de aplicaciones escritas en forma independiente que cooperan para funcionar como un sistema integrado y permiten la adición de nuevas funcionalidades por parte de organizaciones independientes, luego de ser liberado.
Productividad	La productividad es una cualidad del proceso de producción de software, mide la eficiencia del proceso y es una cualidad de desempeño aplicada al proceso. Un proceso eficiente genera una entrega más rápida del producto.
Oportunidad	La oportunidad es la cualidad del proceso que se refiere a la habilidad de entregar un producto a tiempo. Históricamente, los procesos de producción de software no han tenido esta cualidad, lo cual causó la llamada “crisis del software” (que, a su vez, trajo aparejada la necesidad y el nacimiento de la ingeniería de software). Incluso actualmente muchos procesos fracasan en lograr sus resultados a tiempo.

	Entregar un producto a tiempo requiere una planeación cuidadosa, con un trabajo de estimación acertado y puntos de revisión especificados claramente y verificables. Todas las demás disciplinas de la ingeniería utilizan técnicas estándares de administración de proyectos.
Visibilidad	Un proceso de desarrollo de software es visible si todos sus pasos y su estado actual son claramente documentados. Otros términos utilizados para la visibilidad son los de “transparencia” y “apertura”. La idea es que los pasos y el estado del proyecto estén disponibles y sean fácilmente accesibles para ser examinados externamente.

Cualidades de Software como Requerimientos (**FIG, 2011**)

2.4.7. Técnicas para la especificación de Requerimientos

Es importante mencionar que las técnicas que se presentan a continuación no son las únicas que existen:

2.4.7.1. Árboles de decisión

El árbol de decisión es un diagrama que representa en forma secuencial las condiciones y acciones: muestra cuáles condiciones se consideran en primer lugar, cuáles en segundo y así sucesivamente. Este método expone la relación existente entre cada condición y el grupo de acciones permitidas asociadas con ella.

Los árboles de decisión son normalmente construidos a partir de la descripción escrita de un problema y proporcionan una visión gráfica de la toma de decisión necesaria, especifican las variables que son evaluadas y las acciones que deben ser tomadas, así como el orden en el que será efectuada la toma de decisión.

Cada vez que se ejecuta un árbol de decisión sólo un camino será seguido, dependiendo del valor actual de la variable evaluada. Se recomienda el uso del árbol de decisión cuando el número de acciones es pequeño y no son posibles todas las combinaciones.

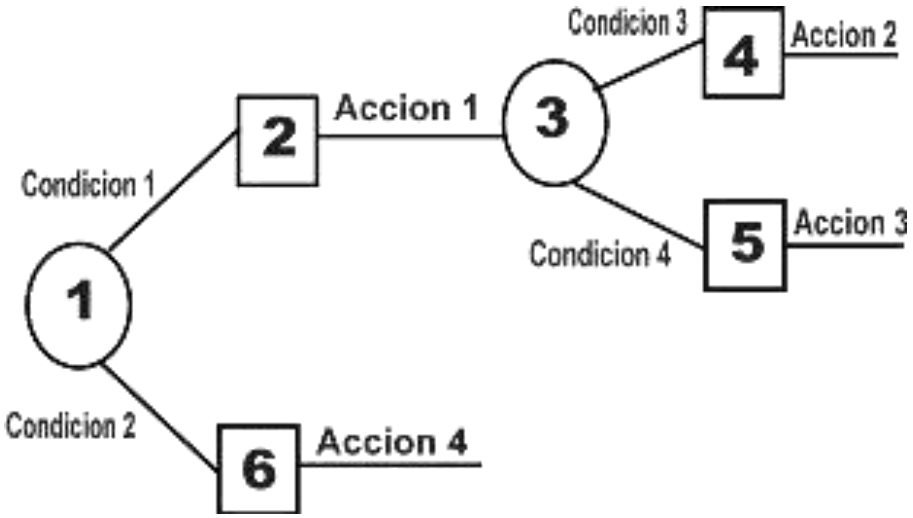


Tabla de decisión: la tabla de decisión es una matriz de filas y columnas que indican condiciones y acciones. Las reglas de decisiones, incluidas en una tabla de decisión, establecen el procedimiento a seguir cuando existen ciertas condiciones. Este método se emplea desde mediados de la década de los 50, cuando fue desarrollado por *General Electric* para el análisis de algunas funciones de la empresa, como el control de inventarios, análisis de ventas, análisis de créditos y control de transporte y rutas. La tabla de decisión es usada cuando existen muchas combinaciones. (Cf. MT, **Tablas de decisión**)

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

2.4.7.2. Español estructurado

También conocido como “lenguaje estructurado”, es el más utilizado para realizar especificaciones de procesos. Es un subconjunto del español, como lo es el inglés en los lenguajes de programación.

El propósito del español estructurado es proporcionar los elementos suficientes para construir especificaciones de procesos tales que, sin perder la capacidad de comunicación que tiene el lenguaje natural, se encaminen hacia lo preciso y lo estructurado de los lenguajes de programación, es decir, se busca especificar el proceso sin perder de vista que lo que se está especificando en la fase de análisis. El esquema de especificación del español estructurado está compuesto de sentencias declarativas simples, estructuras de decisión y estructuras de repetición; también contempla cualquier combinación que pueda darse entre estos esquemas.

En la siguiente tabla te presentamos el vocabulario del español estructurado, el cual está conformado por:

```
si condición entonces
bloque acciones
sino
bloque acciones
fin-si

mientras condición hacer
bloque acciones
fin-mientras

repetir
bloque acciones
hasta condición

para ver desde inicio hasta fin hacer
bloque acciones
fin-para
```

Recomendaciones para especificar con español estructurado:

1. Considerar sólo aquellas palabras que tienen un significado único o, de lo contrario, no serán de ayuda para la descripción de procesos al permitir el surgimiento de ambigüedades.
2. Excluir sinónimos, adjetivos, adverbios y símbolos de exclamación e interrogación.
3. Considerar el uso de modos verbales en infinitivo, como “calcular”, o en imperativo, como “comienza”.
4. Considerar que los verbos deben ser, principalmente, verbos orientados a la acción, como “encontrar”, “obtener”, “sumar”, “reemplazar”, etcétera.

5. Evitar, en lo posible, verbos como “procesar” o “manejar”, ya que se les considera *no significativos* o *redundantes*.
6. Sustituir toda sentencia que pueda reemplazarse por otra más simple.
7. Evitar incluir anotaciones fuera de línea, como notas al margen.

Con estas consideraciones el español estructurado permite configurar las frases de las especificaciones de los procesos. En las frases deben tomarse los nombres dados en los almacenes, flujos de datos y datos elementales en el Diccionario de datos o en el Diagrama de flujo de datos, o bien, deben ser términos locales.

2.4.7.3. Diagramas de Flujo

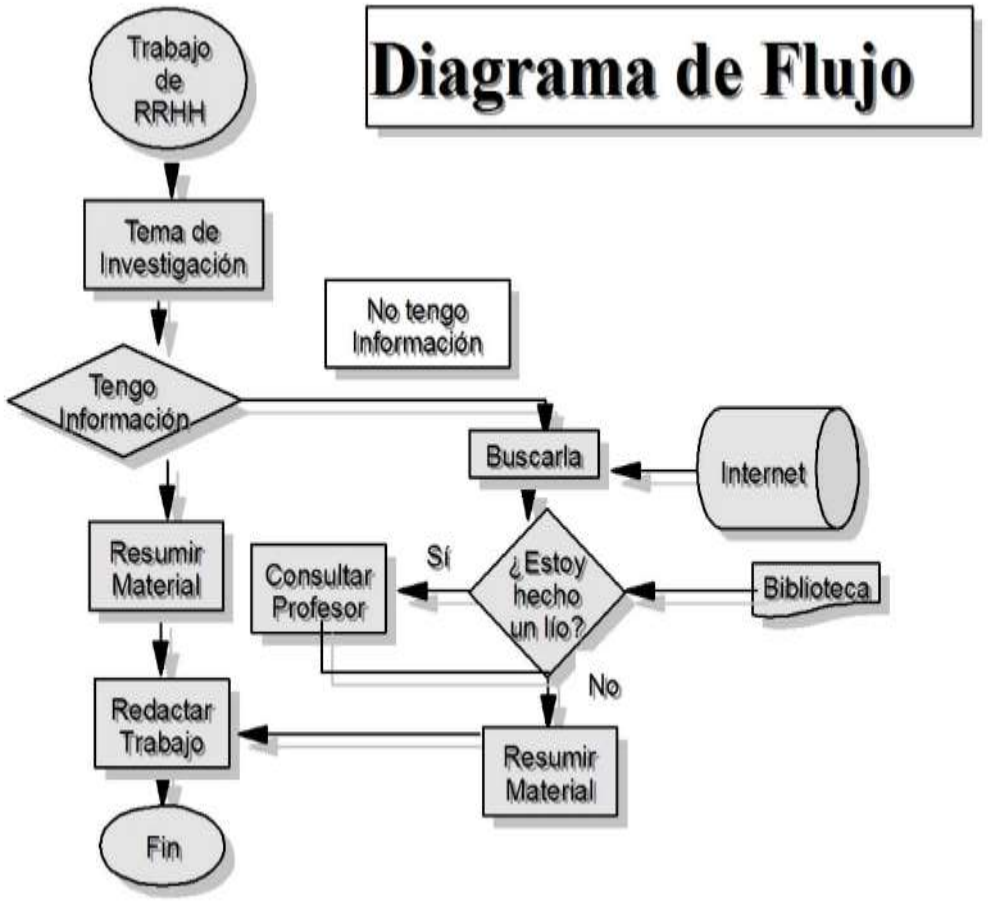
Un *diagrama de flujo* es una forma de representar gráficamente los detalles algorítmicos de un proceso. En la etapa de análisis se ocupan para especificar los procesos que se ejecutan actualmente. Estos diagramas utilizan una serie de símbolos con significados especiales y son la representación gráfica de los pasos de un proceso.

A continuación se lista una serie de acciones que son recomendables hacer antes de elaborar un diagrama de flujo:

• **Identificar a los participantes de la reunión donde se desarrollará el diagrama de flujo. Lo recomendable es que esté presente toda aquella persona que tenga que ver o se vea afectada por el proceso.**
• **Definir qué se espera obtener del diagrama de flujo.**
• **Identificar quién lo empleará y cómo.**
• **Establecer el nivel de detalle requerido.**
• **Determinar los límites del proceso por describir**

Los pasos para elaborar el diagrama de flujo son:

• Establecer el alcance del proceso por describir; de esta manera quedará fijado el comienzo y el final del diagrama. Frecuentemente el comienzo es la salida del proceso previo y el final la entrada al proceso siguiente.
• Identificar y enlistar las principales actividades que están incluidas en el proceso que se va a describir, así como su orden cronológico.
• Identificar y enlistar los puntos de decisión.
• Construir el diagrama respetando la secuencia cronológica y asignando los correspondientes símbolos.
• Asignar un título al diagrama y verificar que esté completo y describa con exactitud el proceso elegido.



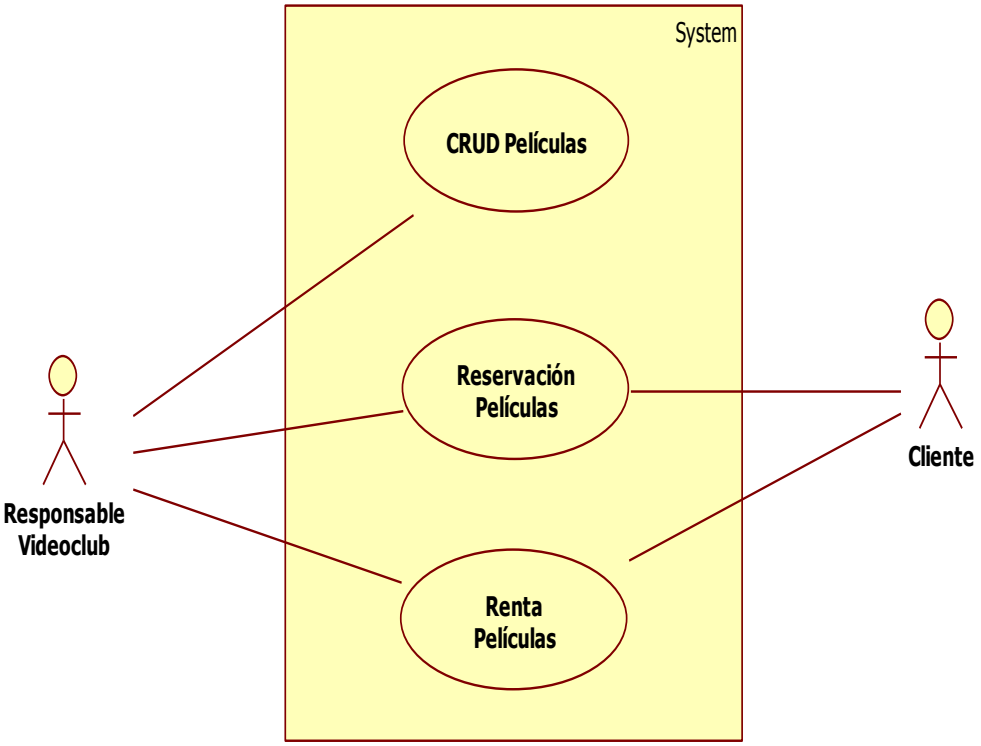
2.4.7.4. Especificación de casos de uso

Nótese que la numeración es un factor importante para la claridad de la redacción, ya que nos permite describir situaciones donde el actor tiene que elegir una opción de un conjunto de alternativas. Las postcondiciones son condiciones o acciones que se tienen que hacer después de concluidos alguno de los flujos del caso de uso.
Las excepciones ayudan a establecer flujos que están fuera de lo normal, es decir, son eventos que no tienen nada que ver con las actividades normales del negocio. Ejemplo de ello, tenemos que el usuario puso texto en un campo que se esperan números o que haya fallado la conexión a la base de datos, que se haya ido la luz, que el protocolo “http” (*hyper text transfer protocol*) no esté disponible, etcétera.

En el Modelo de Casos de uso se utilizan los diagramas de casos de uso en los que participan los actores y los casos de uso. La característica principal de este modelo es que identifica requerimientos de tipo funcional. Los actores pueden ser personas, organizaciones u otro software que interactúa con el sistema que estamos modelando. Para designarlos se tiene que poner el nombre del rol que está jugando ese actor con respecto al sistema.

Los casos de uso representan la funcionalidad que observan directamente los actores y especifican los requerimientos funcionales. Asimismo, son procesos que lleva a cabo el sistema para procesar los datos proporcionados por los actores y generar la información que ellos requieren. Su símbolo es una elipse u óvalo con el nombre del caso de uso dentro de él o debajo de él.

Los actores y casos de uso se relacionan entre sí con una línea recta que se llama asociación. Esta asociación indica que hay un intercambio de datos/información entre el actor y el caso de uso. Se recomienda poner los casos de uso dentro de un rectángulo para indicar los límites del sistema; este rectángulo va acompañado de una leyenda que dice “system” en la esquina superior derecha.



Documento de Especificaciones Técnicas

El diagrama de casos de uso no es suficiente para entender qué se necesita que haga el sistema, por lo que se hace pertinente describir el detalle de cada caso de uso. Esta descripción también recibe el nombre de especificación de caso de uso o especificación técnica.

Las secciones de una descripción básica de un caso de uso son:

- Nombre
- Descripción
- Actores
- Precondiciones
- Flujo principal o normal

- Flujos alternativos
- Postcondiciones
- Excepciones

A continuación te explicaremos cada una.

Nombre

El **nombre** corresponde al caso de uso que se desea especificar del diagrama de casos de uso. Es necesario recalcar que el nombre del caso de uso empieza con un verbo y después lo acompañan los sustantivos que permitan identificar qué es lo que hace.

Descripción

La **descripción** es texto breve que explica en qué consiste el caso de uso.

Actores

Los **actores** son los que intervienen en ese caso de uso, por lo que sólo se deben mencionar los actores que están asociados con el caso de uso en el diagrama.

Precondiciones

Las **precondiciones** son condiciones que se pueden o se deben presentar antes de que inicie el caso de uso.

La parte más importante se conforma del **flujo principal** y de los **flujos alternativos**: en algunas ocasiones se juntan o se separan, todo depende de cuál escritura facilita más su comprensión.

La redacción de los casos de uso debe ser simple, ya que es una narrativa que describe las acciones que hay entre el sistema y el actor. Siempre tiene el patrón

“el actor hace, el sistema hace”, “el actor hace, el sistema hace”. Una ayuda de esto es la numeración de cada una de las actividades que ocurren entre el actor y el sistema.

Hay que tener cuidado de evitar describir interrelaciones entre actores: lo único que interesa son las interrelaciones entre el actor y el sistema.

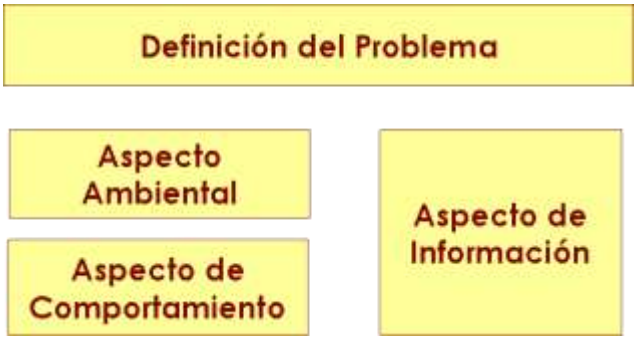
2.5. Modelado del análisis

Cuando se requiere comunicar los resultados de abstraer la realidad, respecto a cierto ambiente, no es suficiente que se expresen en forma verbal sino que se hace uso de los modelos. Un modelo es una representación de la realidad: de acuerdo con el caso del análisis de sistemas y el paradigma que se esté utilizando, se emplean ciertos diagramas para especificar los modelos.

La comunicación se da en forma interna o externa, según el equipo de desarrollo. La comunicación interna requiere de mayor especificidad por tratarse de iguales en el equipo de desarrollo, al contrario de la comunicación externa, cuya formalidad puede ser menor (a menos que sea un requerimiento por parte del cliente). En la comunicación externa se requiere de simplicidad y formalidad, los usuarios y clientes desconocen los tecnicismos del proceso de desarrollo de software, por lo que se debe tener especial cuidado de elegir los artefactos que se presentarán al usuario e, incluso, se tendría que capacitar a los usuarios en cuanto al uso de éstos.

Tras identificar el problema por resolver, se procede a documentar la parte dinámica y estática del sistema actual. Los aspectos que se encargan de modelar la parte dinámica del sistema son el Ambiental y el de Comportamiento. El aspecto que se encarga de modelar la parte estática del sistema es el Aspecto de Información que

va a generar un modelo, por lo que al final tendremos un Modelo Ambiental, un Modelo de Comportamiento y un Modelo de Datos.



Aspectos del Modelo de Análisis

¿Por cuál aspecto se debe comenzar? La respuesta está en función de los datos que tenemos a nuestra disposición y de la comprensión de tales datos.

Si tenemos un mejor conocimiento de los procesos de la organización, es recomendable empezar por el Aspecto Ambiental y el Aspecto de Comportamiento para luego pasar al Aspecto de Información. Si tenemos un mejor conocimiento sobre los datos e información que genera la empresa, es recomendable empezar por el Aspecto de Información y posteriormente con el Aspecto Ambiental y el Aspecto de Comportamiento. A estos aspectos se les denomina “elementos del modelo de análisis de sistemas”.

2.5.1. Modelado Ambiental

El objetivo del aspecto ambiental es modelar el aspecto dinámico externo del sistema, es decir, establecer tanto el alcance del sistema como sus límites. Para modelar este aspecto se utiliza el “diagrama de contexto”, el cual se utiliza para el sistema actual y para el sistema propuesto, apoyándose de tres símbolos:

2.5.1.1. Diagramas de Contexto (DC)

Nombre	Descripción
Burbuja (Sistema) 	Representa al sistema que se está modelando y sólo hay una burbuja en cada diagrama de contexto. Dentro de ella se debe poner el nombre del sistema que se está representando.
Rectángulo (Entidad) 	Representa un agente externo o entidad que interactúa con el sistema, proporcionando datos o recibiendo información del sistema. Dentro del rectángulo se debe poner el nombre del agente externo con mayúsculas y en singular. Los agentes externos pueden ser personas, organizaciones u otro sistema.
Flechas (Flujos) 	Representan el sentido en que se envía la información, ya sea del sistema a un agente externo o en sentido contrario. Hay de dos tipos de sentido: *unidireccionales* y *bidireccionales*. • En los casos unidireccionales sólo hay una flecha en un extremo que se ocupa para representar el envío o la recepción de datos de manera asíncrona. • Los casos bidireccionales tienen flechas en ambos extremos y representan que se están enviando y recibiendo datos de manera síncrona. Se pueden nombrar en forma de título o con minúsculas y siempre deben estar nombrados con sustantivos, no con verbos; además, se deben omitir las palabras “datos” e “información”, ya que su uso sería redundante. Al final de cuentas, en este diagrama lo que se quiere ver son los flujos de información entre el sistema y los agentes externos.

Símbolos del Diagrama de Contexto

En el cuadro anterior puedes observar por el nombre que tiene la burbuja que se trata de un sistema de un videoclub, el cual se lleva de manera manual. Realmente es un sistema muy sencillo donde el Responsable del Videoclub envía al sistema los datos de la película y del cliente para realizar la renta y, como respuesta, el Responsable del Videoclub tiene un comprobante de Renta. De igual manera, por medio del título de la película o del autor, puede buscar las películas y obtendrá un listado de las que coincidan con esos datos. Por último, el Responsable del Videoclub proporciona los datos de la película, ya sea para ingresar una nueva y actualizar o para eliminar una película existente.

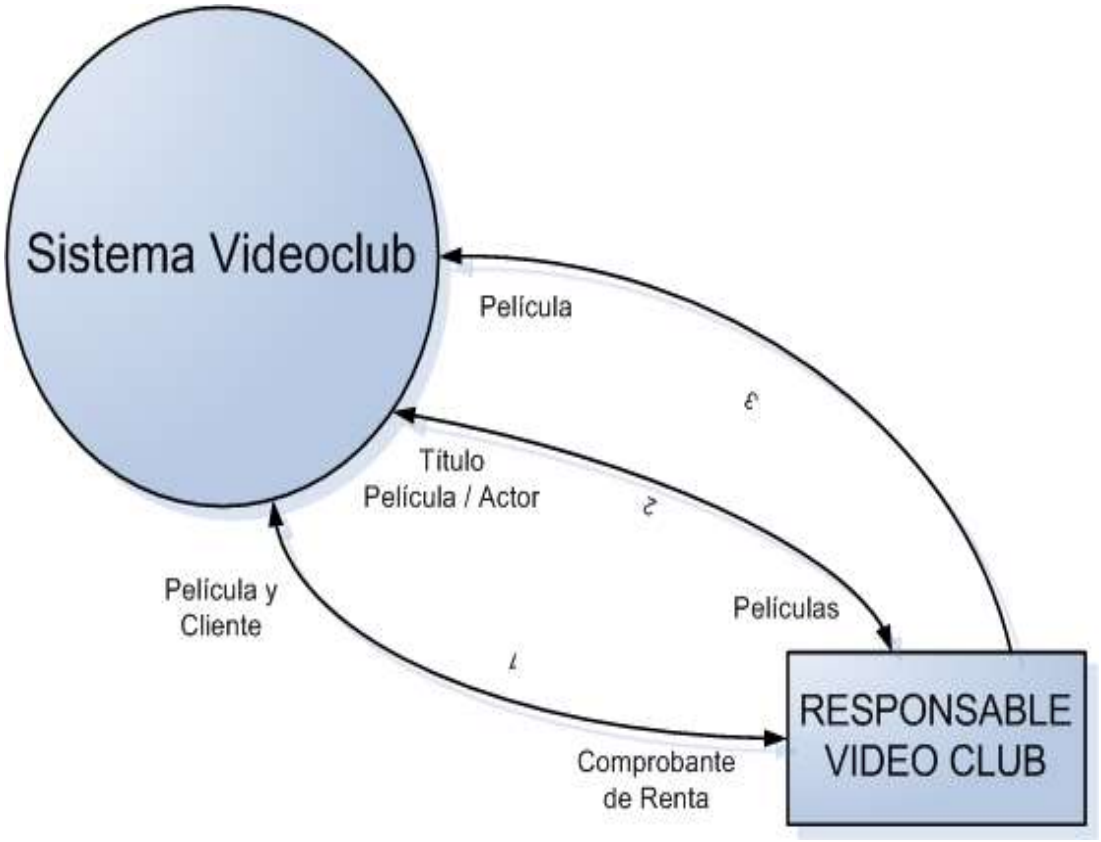


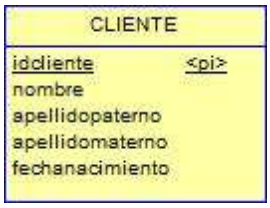
Diagrama de Contexto

2.5.2. Modelado de datos

El objetivo de la información es modelar el aspecto estático del sistema, es decir, las estructuras de persistencia de datos y las relaciones que existen entre ellas. Para modelar el aspecto de información se utiliza el Diagrama de Entidad-Relación (DER); cabe mencionar que este diagrama se utiliza tanto para el sistema actual como para el sistema propuesto.

2.5.2.1. Diagrama Entidad-Relación (DER)

Las entidades son los elementos más significativos en el DER, cada entidad está compuesta por atributos. En un futuro, cuando el DER se convierta en una base de datos, cada atributo contendrá un tipo de datos:

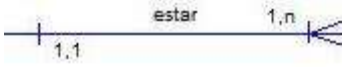
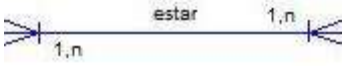
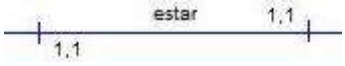
Nombre	Descripción
Rectángulo (Entidad) 	Representa una relación, también llamada entidad, la cual almacena datos de un objeto de la vida real. Las entidades son los almacenes que se emplean en el Diagrama de flujo de datos. Dentro del rectángulo se debe poner el nombre de la entidad con mayúsculas y en singular. También se deben poner los atributos de la entidad, es decir, sus características, en minúsculas y sin espacios. Hay un atributo especial denominado *identificador*, el cual tiene como propósito distinguir un registro específico con respecto a los demás. En algunos casos el identificador es compuesto, es decir, está formado por más de un atributo.

Relación 	Representa una relación bidireccional entre dos entidades. Las relaciones se dibujan con una línea continua con su nombre en minúsculas. El nombre de la relación debe estar escrito con un verbo y al centro de la línea. Otro elemento importante de las relaciones es la "cardinalidad", la cual indica la forma en que se relacionan las entidades.

Símbolos del Diagrama de Entidad-Relación

Cardinalidad y Modalidad

Existen tres tipos de *cardinalidad*: de uno a uno, de uno a muchos y de muchos a muchos. Observa la siguiente tabla.

Símbolo	Descripción
	Relación de uno a muchos
	Relación de muchos a muchos
	Relación de uno a uno

Símbolos para representar la Cardinalidad

Hay cuatro tipos de *modalidad*:

Modalidad	Descripción
0, 1	Establece una interrelación en donde la entidad puede tener/contener ningún o un elemento de la otra entidad.
1, 1	Establece una interrelación en donde la entidad debe tener/contener un elemento de la otra entidad.
0, n	Establece una interrelación en donde la entidad puede tener/contener ningún o más de un elemento de la otra entidad.

1, n	Establece una interrelación en donde la entidad puede tener/contener uno o más de un elemento de la otra entidad.

Nomenclatura para representar la Modalidad

En la siguiente sección se explicará con un ejemplo cómo se emplean la cardinalidad y la modalidad.

Se hace necesario recalcar que hay dos tipos de diagramas de entidad-relación: el *lógico* y el *físico*. El primero se genera en la fase de análisis y el segundo en la fase de diseño. El siguiente ejemplo representa un diagrama de entidad-relación lógico del “sistema videoclub”.

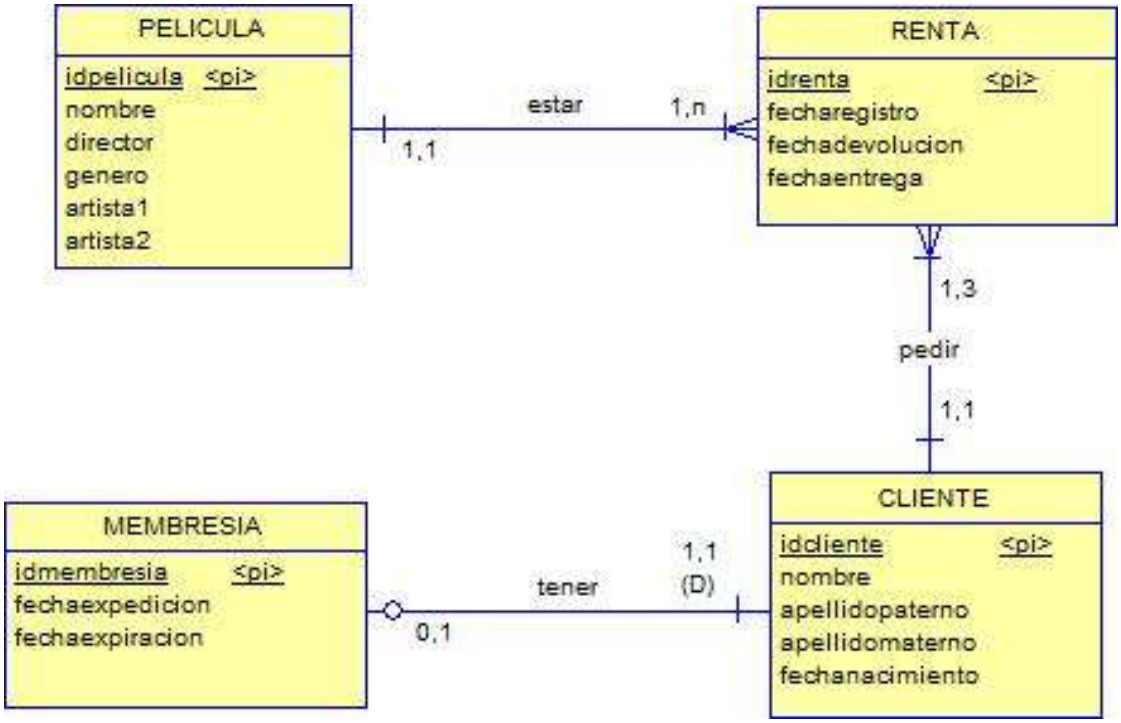


Diagrama de Entidad-Relación

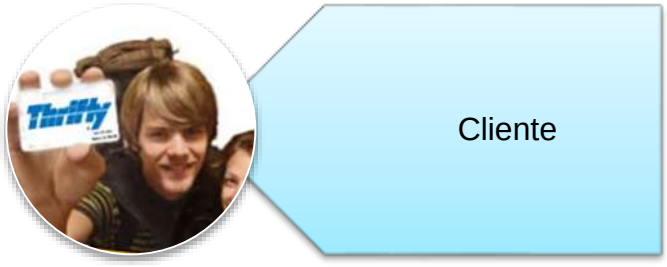
Como primera relación encontramos la de la entidad PELÍCULA con la entidad RENTA, la cual se leería “una película está en una o más rentas, y una renta está o

tiene una película". En esta relación apreciamos que la cardinalidad se pone en ambos extremos:

- La que está del lado izquierdo se lee "una renta está o tiene una película", por eso se pone "1 coma 1", donde el primer número representa el valor mínimo y el segundo número representa el valor máximo, estos números reciben el nombre de modalidad.
- La cardinalidad del lado derecho se lee "una película está en renta una o más veces", por eso se pone "1 coma n", nótese que en este caso tenemos una representación iconográfica denominada "patas de gallo", que sólo se utiliza en las relaciones de uno a muchos o de muchos a muchos.

Otra relación es la que hay entre la entidad MEMBRESÍA y la entidad CLIENTE, la cual se leería "una membrecía es/tiene uno y sólo un cliente, y un cliente puede tener una membresía". La cardinalidad del lado izquierdo se lee "un cliente puede tener una membresía", por eso se pone "0 coma 1". La cardinalidad del lado derecho se lee "una membresía es/tiene uno y solo un cliente", por eso se pone "1 coma 1".

En este tipo de relación es importante denotar la entidad que es dominante, para ello se indica con la letra "D" que la entidad *cliente* es la fuerte; para cuando se pase al *diagrama de entidad relación física,* la entidad "cliente" pasará su llave primaria como foránea a la entidad "membrecía" y con eso se evita tener referencia circular entre tablas.



Con estos dos ejemplos de relaciones se recalca la importancia de identificar las reglas de negocio en la fase de requerimientos, ya que ellas nos indican cómo construir esos diagramas.

2.5.2.2. Diccionario de Datos

Los diagramas utilizados para modelar los aspectos dinámicos y estáticos del sistema no son suficientes para que diseñadores y programadores puedan continuar con su tarea. Si el analista sólo considera el nombre de los flujos de datos, dejaremos a su imaginación lo que contienen, causando problemas en el diseño y las fases subsecuentes. Cada flecha de un DFD representa uno o más elementos de información, por lo tanto, el analista debe disponer de algún otro método para representar el contenido de cada flecha. El *diccionario de datos* es una herramienta que define una gramática para describir el contenido de los elementos de información, tales como los siguientes:

Carácter(es)	Descripción
=	está compuesto de
+	y
( )	optativo (puede estar presente o ausente)
{ }	Iteración
[ ]	seleccionar una de varias alternativas
* *	comentario
@	identificador (campo clave) para un almacén
|	separa opciones alternativas en la construcción

Símbolos del Diccionario de Datos

Basándonos en el ejemplo del videoclub, el uso de algunos elementos anteriores

renta =	membresía + nombre + (segundo nombre) + apellido paterno + apellido materno
título de cortesía =	[Oro | Plata | Bronce]
nombre =	{carácter legal}
apellido paterno =	{carácter legal}
apellido materno =	{carácter legal}
carácter legal =	{A-Za-z}

El diccionario de datos debe contener, en la medida de lo posible y lo valioso, las definiciones de todos los datos mencionados en el DFD. Los datos compuestos (datos que pueden ser divididos, como el nombre completo de una persona) se definen en términos de sus componentes; los datos elementales (datos que no pueden ser divididos) se definen en términos del significado de cada uno de los valores que puede asumir.

Este diccionario es un listado organizado de los datos pertinentes al sistema, con definiciones exentas de ambigüedades para que, tanto el usuario como el analista, tenga un entendimiento común de todas las entradas, salidas y componentes de los almacenes.

El diccionario de datos no solo sirve para especificar los flujos de datos sino también para especificar cada almacén contenido en los DFD. Al realizar esta actividad se especifican las entidades que conforman el diagrama de entidad-relación: ¿por qué? porque a cada almacén debe corresponder una entidad en el DER.

2.5.3. Modelado de Comportamiento

El objetivo del comportamiento es modelar el aspecto dinámico interno del sistema. Para modelar este aspecto se utiliza el Diagrama de Flujo de Datos, el cual modela el aspecto dinámico tanto de un sistema manual como de un sistema automatizado, igual que lo hace un Diagrama de Contexto.

2.5.3.1. Diagrama de Flujo de Datos (DFD)

Al igual que en el Diagrama de Contexto, en los Diagramas de Flujo de Datos se emplean los símbolos de una burbuja, que representa al proceso; un rectángulo, que representa al agente externo, y una flecha, que representa al flujo. Estos símbolos tienen la misma semántica, pero se incluye un nuevo símbolo compuesto por dos líneas horizontales paralelas.

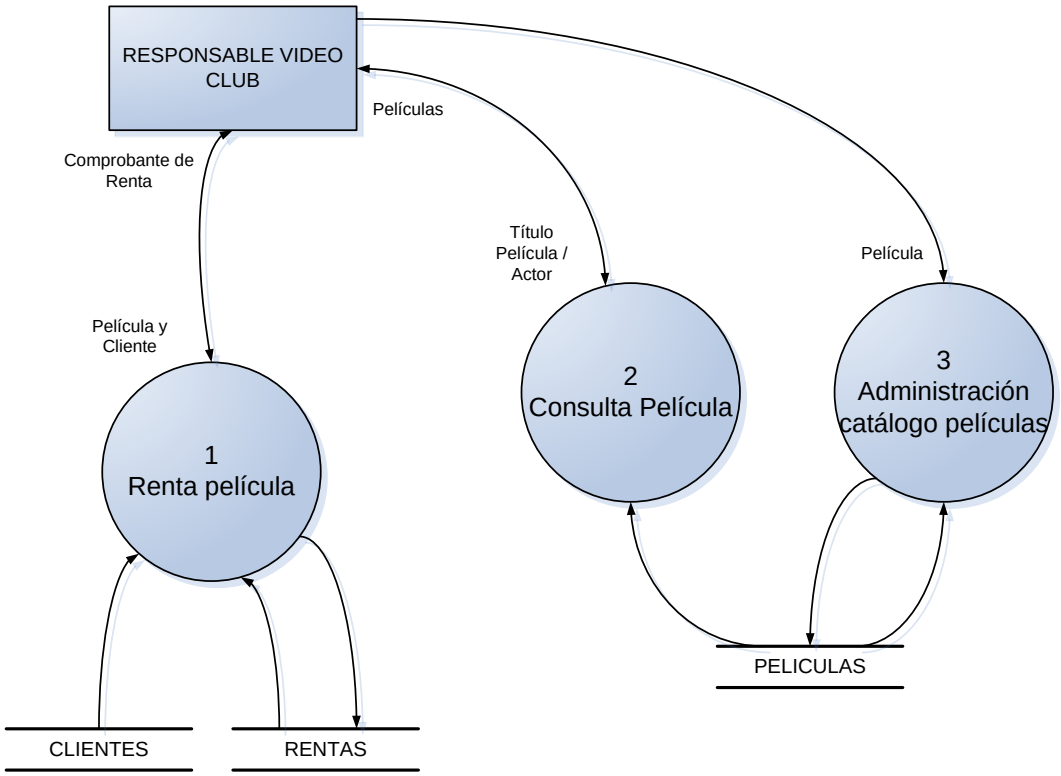


Diagrama de Flujo de Datos

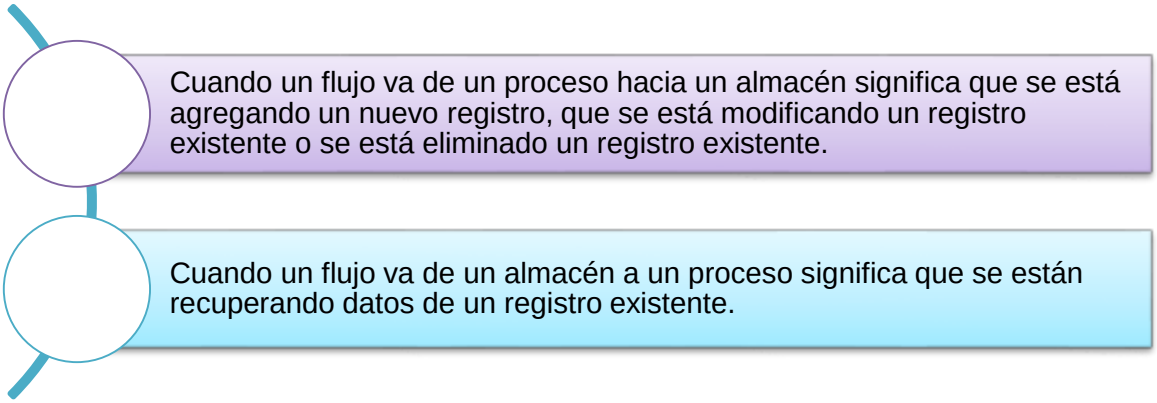
Las líneas horizontales paralelas se utilizan para modelar un almacén (los almacenes son repositorios de datos). Su objetivo es que se pueda guardar, modificar, eliminar y consultar datos cuando se requiera. En el caso de un sistema manual, un almacén es un archivero, agenda o cualquier lugar donde se almacenen papeles con datos. En el caso de un sistema automatizado, los almacenes son archivos electrónicos o tablas en alguna base de datos. Los nombres de los almacenes deben ser sustantivos y se deben escribir con mayúsculas y en plural.

Los diagramas de contexto no son suficientes para tener una visión completa del sistema sino que falta conocer cómo trabaja o trabajará internamente. Los DFD ayudan en esta parte: teniendo un diagrama de contexto, se puede proceder a crear su DFD correspondiente (Diagrama de Flujo de Datos de Nivel 0 (cero) ó 1). En este diagrama se descompone el sistema en sus procesos principales, pero regladamente sólo puede haber 7+2 procesos, es decir, hasta nueve procesos como máximo. Cuando se rebasan los nueve procesos significa que se está siendo muy detallado, por lo que será necesario juntar uno o más procesos. No es correcto tener un diagrama de flujo de datos que sólo tenga una burbuja.

En los DFD se busca identificar los procesos que reciben datos, los procesan y generan información. En este caso, las burbujas no representan sistemas sino procesos que deben ir numerados, aunque esa numeración NO indica su precedencia, sólo se utiliza para identificar un proceso de otro. Todo proceso debe iniciar con un verbo. Los procesos solo pueden tener flujos de un agente externo o de un almacén. Asimismo, sólo se puede comunicar un proceso con otro cuando se trata de un sistema en tiempo real. El sentido de la dirección de los flujos indica si el proceso recibe o proporciona datos.

Los flujos que van o vienen de un almacén no llevan nombre porque sólo se puede enviar u obtener datos que correspondan al almacén, esto hace necesario que se

tenga un nombre adecuado para los datos que guarda. El significado del flujo varía según la dirección que tenga:



Los procesos de los DFD pueden dividirse, a su vez, en otro DFD, el cual está conformado por los subprocesos que conforman el proceso principal. ¿Hasta qué profundidad se puede hacer esto? Lo más recomendable es no pasar de los siete niveles y se siguen las mismas reglas que para los DFD de Nivel 1.

En este ejemplo se puede apreciar que descompusimos el “Proceso de Administración de Catálogo de Películas” y que los procesos tienen otra numeración, la cual está formada por el número del “proceso padre” y el número consecutivo de cada proceso.

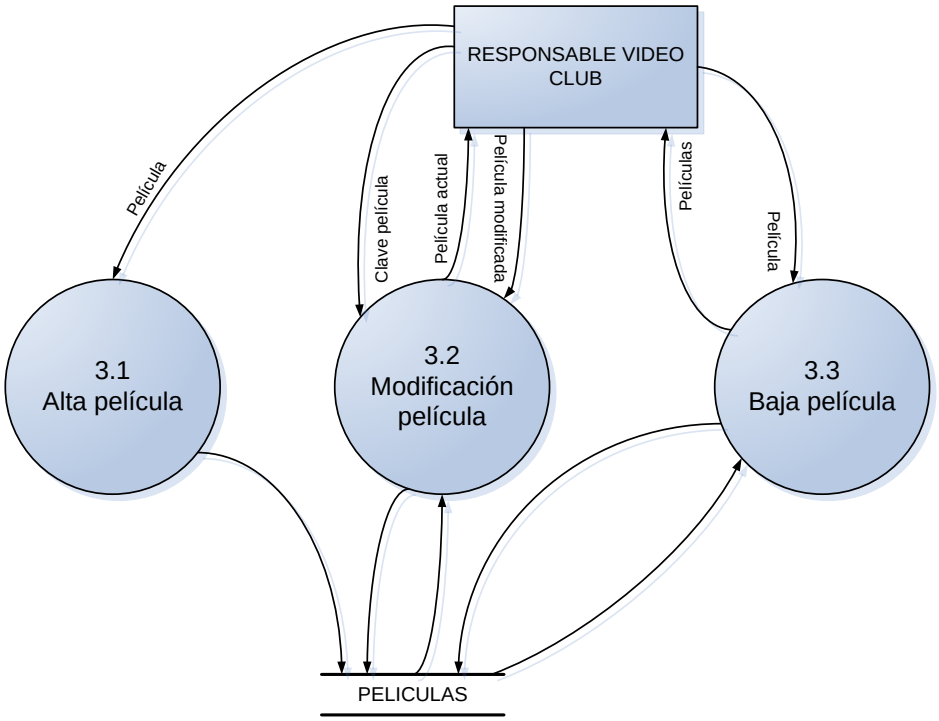
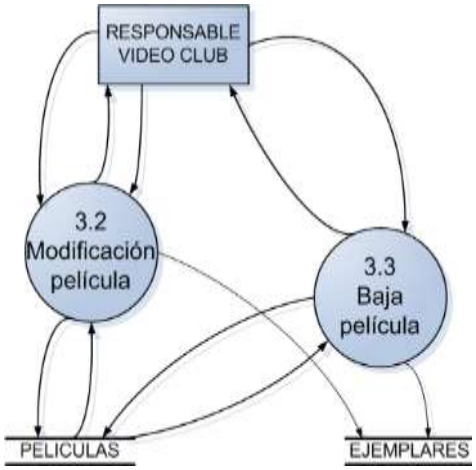


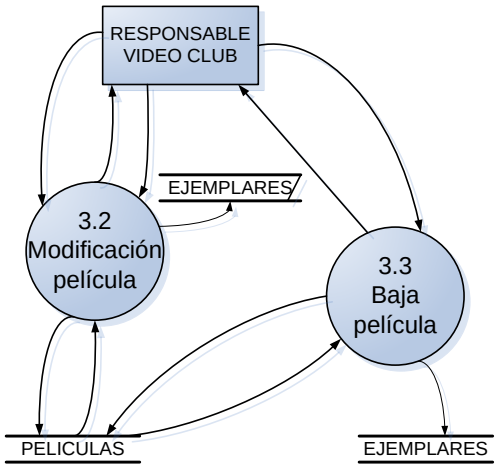
Diagrama de Flujo de Datos de Nivel 2

En algunas situaciones puede presentarse un flujo que atraviesa otros flujos, lo cual no es recomendable porque dificulta la lectura del diagrama y puede llegar a convertirse en una telaraña. Para estos casos, lo recomendable es repetir la entidad o almacén que está involucrada, agregando una línea inclinada en la parte inferior derecha para indiciar que esa entidad o almacén ya fue creado anteriormente y sólo se trata de un clon de la original.

Incorrecto



Correcto



Diagramas de Flujo de Datos: Forma Incorrecta y Correcta

2.5.3.2. Diagrama de Transición de Estados (DTE)

Otro diagrama que se utiliza para modelar el aspecto dinámico interno del sistema es el Diagrama de Transición de Estados (DTE). En este diagrama se busca representar los diferentes estados que puede tener una entidad del Diagrama de Entidad-Relación. La simbología que se utiliza es la siguiente:

Nombre	Descripción
Rectángulo (Estado) RENTA Realizada	Representa un estado que puede tener la entidad, ya sea por un evento de información o por uno de tiempo. Dentro del rectángulo se debe poner el nombre del estado a manera de buena práctica, para lograr una mejor comprensión

Rectángulo vacío (Estado final) 	Representa un estado final, es decir, es cuando la entidad deja de existir. En un ambiente de producción significa que el registro u objeto es borrado físicamente, por lo que no podrá ser recuperado.
Flechas (Evento de transición) 	Representan los eventos de información o eventos de tiempo que provocan un cambio de estado en la entidad. Las flechas están acompañadas del nombre del evento que cambia el estado de la entidad. Hay tres clasificadores para los eventos: **C:** para cuando el evento *crea* la entidad. **M:** para cuando el evento *modifica* o actualiza el estado de la entidad. **E:** para cuando el evento *elimina* la entidad, lo cual implica que la entidad ya no existirá en la base de datos.

Símbolos del Diagrama de Transición de Estados

Cabe mencionar que los estados de las entidades están conformados por los valores que tienen las entidades en un momento dado.

En el ejemplo de la siguiente figura se muestran los diferentes estados por los que atraviesa la entidad RENTA, al principio. Por el evento “El cliente se presenta en el mostrador con las películas que quiere rentar” se crea la entidad RENTA y tiene el estado de “Realizada”; el evento “El cliente agrega otra película a su renta” modifica el estado de la RENTA, y el evento “El responsable del Videoclub actualiza la renta” lo regresa a su estado original.

La entidad RENTA puede pasar al estado de “Cancelada” por el evento “El cliente cancela su renta” o por el evento “El responsable del Videoclub cancela la renta por sanción del cliente” en este caso vemos que es un estado final y, por último, vemos

el evento “Es tiempo de depurar los registros de hace 10 años” que genera un estado especial, representado por un rectángulo vacío, lo cual significa que la entidad es destruida físicamente de la base de datos.

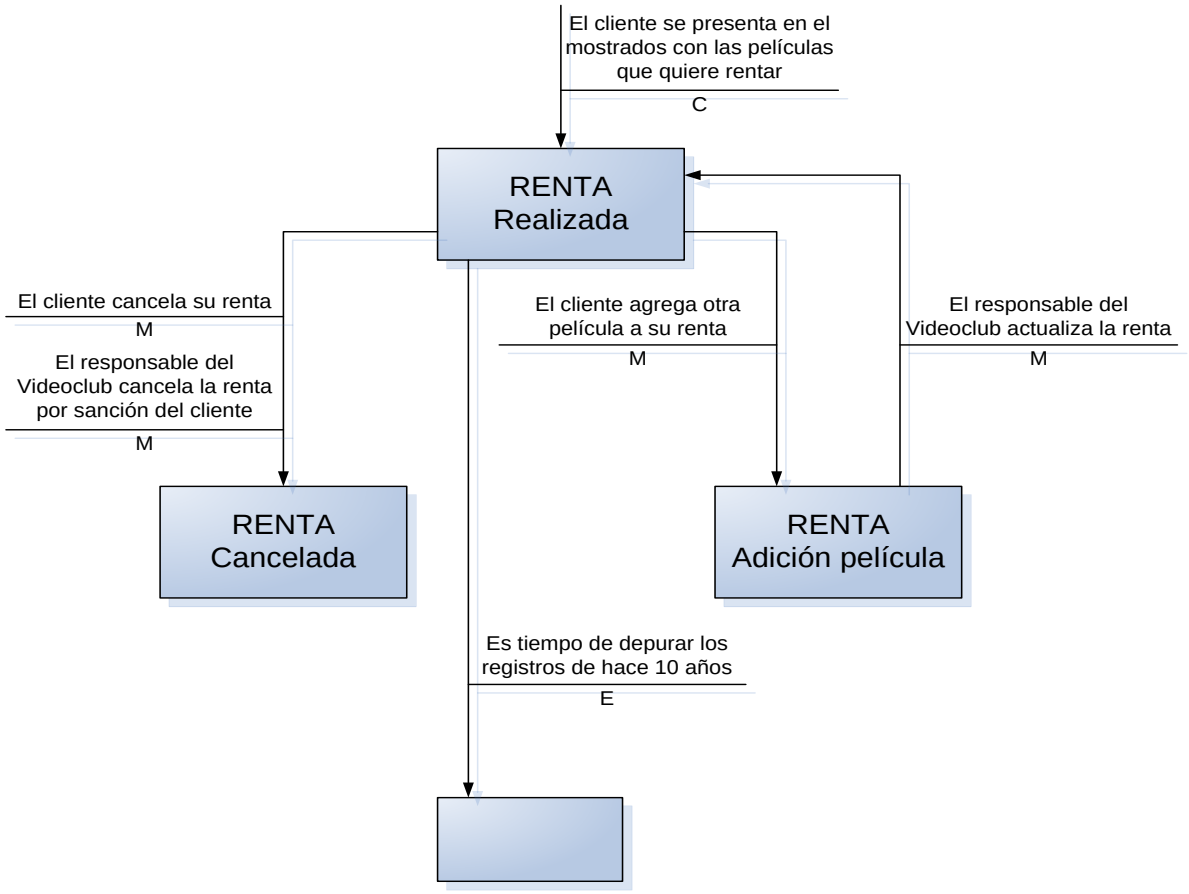


Diagrama de Transición de Estados

2.5.3.3. Diccionario de Datos

Los diagramas utilizados para modelar los aspectos dinámicos y estáticos del sistema no son suficientes para que diseñadores y programadores puedan continuar con su tarea. Si el analista sólo considera el nombre de los flujos de datos, dejaremos a su imaginación lo que contienen, causando problemas en el diseño y las fases subsecuentes.

Cada flecha de un DFD representa uno o más elementos de información, por lo tanto, el analista debe disponer de algún otro método para representar el contenido de cada flecha. El *diccionario de datos* es una herramienta que define una gramática para describir el contenido de los elementos de información, tales como los siguientes:

Carácter(es)	Descripción
=	está compuesto de
+	y
( )	optativo (puede estar presente o ausente)
{ }	Iteración
[ ]	seleccionar una de varias alternativas
* *	comentario
@	identificador (campo clave) para un almacén
|	separa opciones alternativas en la construcción

Símbolos del Diccionario de Datos

Basándonos en el ejemplo del videoclub, el uso de algunos elementos anteriores sería de la siguiente manera:

renta = membresía + nombre + (segundo nombre) + apellido paterno + apellido materno

título de cortesía = [Oro | Plata | Bronce]

nombre = {carácter legal}

apellido paterno = {carácter legal}

apellido materno = {carácter legal}

carácter legal = {A-Za-z}

El diccionario de datos debe contener, en la medida de lo posible y lo valioso, las definiciones de todos los datos mencionados en el DFD. Los datos compuestos (datos que pueden ser divididos, como el nombre completo de una persona) se definen en términos de sus componentes; los datos elementales (datos que no pueden ser divididos) se definen en términos del significado de cada uno de los valores que puede asumir.

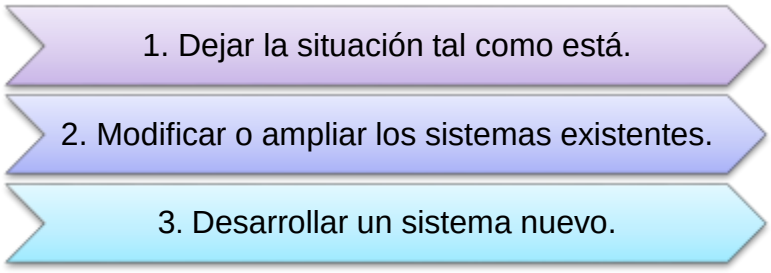
Este diccionario es un listado organizado de los datos pertinentes al sistema, con definiciones exentas de ambigüedades para que, tanto el usuario como el analista, tenga un entendimiento común de todas las entradas, salidas y componentes de los almacenes.

El diccionario de datos no solo sirve para especificar los flujos de datos sino también para especificar cada almacén contenido en los DFD. Al realizar esta actividad se especifican las entidades que conforman el diagrama de entidad-relación: ¿por qué? porque a cada almacén debe corresponder una entidad en el DER.

2.6. Estudio de factibilidad y análisis costo-beneficio

El análisis de sistemas implica un estudio de factibilidad para determinar si la solución es factible dados los recursos y restricciones de la organización.

Normalmente este proceso evalúa 3 alternativas de solución básicas para cada problema.



Los estudios de factibilidad están definidos en 3 áreas:

Factibilidad técnica.- Determina si la solución propuesta se puede implementar o no con el hardware, software y recursos técnicos disponibles.

Factibilidad operativa.- Determina si la solución propuesta es o no deseable dentro del marco administrativo y de organización.

Factibilidad económica.- Si los beneficios de la solución propuesta superan o no a los costos. (Véase Laudon y Laudon, 2004)

Factibilidad técnica	•Determina si la solución propuesta se puede implementar o no con el hardware, software y recursos técnicos disponibles.
Factibilidad operativa	• Determina si la solución propuesta es o no deseable dentro del marco administrativo y de organización.
Factibilidad económica	•Si los beneficios de la solución propuesta superan o no a los costos. (Véase Laudon y Laudon, 2004)

Identificación de beneficios y costos

Un analista de sistemas tiene muchos métodos para analizar costos y beneficios:

- El análisis de punto de equilibrio examina el costo del sistema actual versus el costo del sistema propuesto.
- El método de análisis del tiempo de recuperación de la inversión (ROI) determina el tiempo que tomará antes de que el nuevo sistema sea aprovechable.
- El análisis de flujo de efectivo es apropiado cuando es crítico saber la cantidad de desembolsos de efectivo, mientras que el análisis de valor presente toma en consideración el costo de pedir prestado el dinero.
- El valor presente neto que es el valor monetario de una inversión, tomando en cuenta su costo, ganancias y valor del dinero en el tiempo.
- Índice de rentabilidad, sirve para comparar la rentabilidad de alternativas de inversión; se calcula dividiendo el valor presente de la entrada total de efectivo de una inversión entre el costo inicial de la inversión.
- Cociente costo/beneficio, método para calcular el rendimiento de un gasto de capital, en el que se dividen los beneficios totales entre los costos totales.

RESUMEN

El análisis es una actividad muy importante, ya que si se hace de manera incorrecta, las demás etapas se harán mal, trayendo como consecuencia que el sistema no cumpla con las necesidades por las que fue creado.

Los actores y casos de uso se relacionan entre sí con una línea recta que se llama *asociación*.

BIBLIOGRAFIA

SUGERIDA

Autor	Capítulos	Páginas
Pressman (2005)	1, 2, 7, 12	24, 105, 135, 205, 214, 227
Braude (2003)	1, 6, 9, 10, 11, 12	30,107-123, 175 – 208, 260
Lawrence (2002)	2, 4, 6, 12	72, 157, 107, 201 - 226
Piattini (2000)	2, 3, 9, 17, 26	26, 136, 157, 352, 561
Pressman (2001)	4	51-77

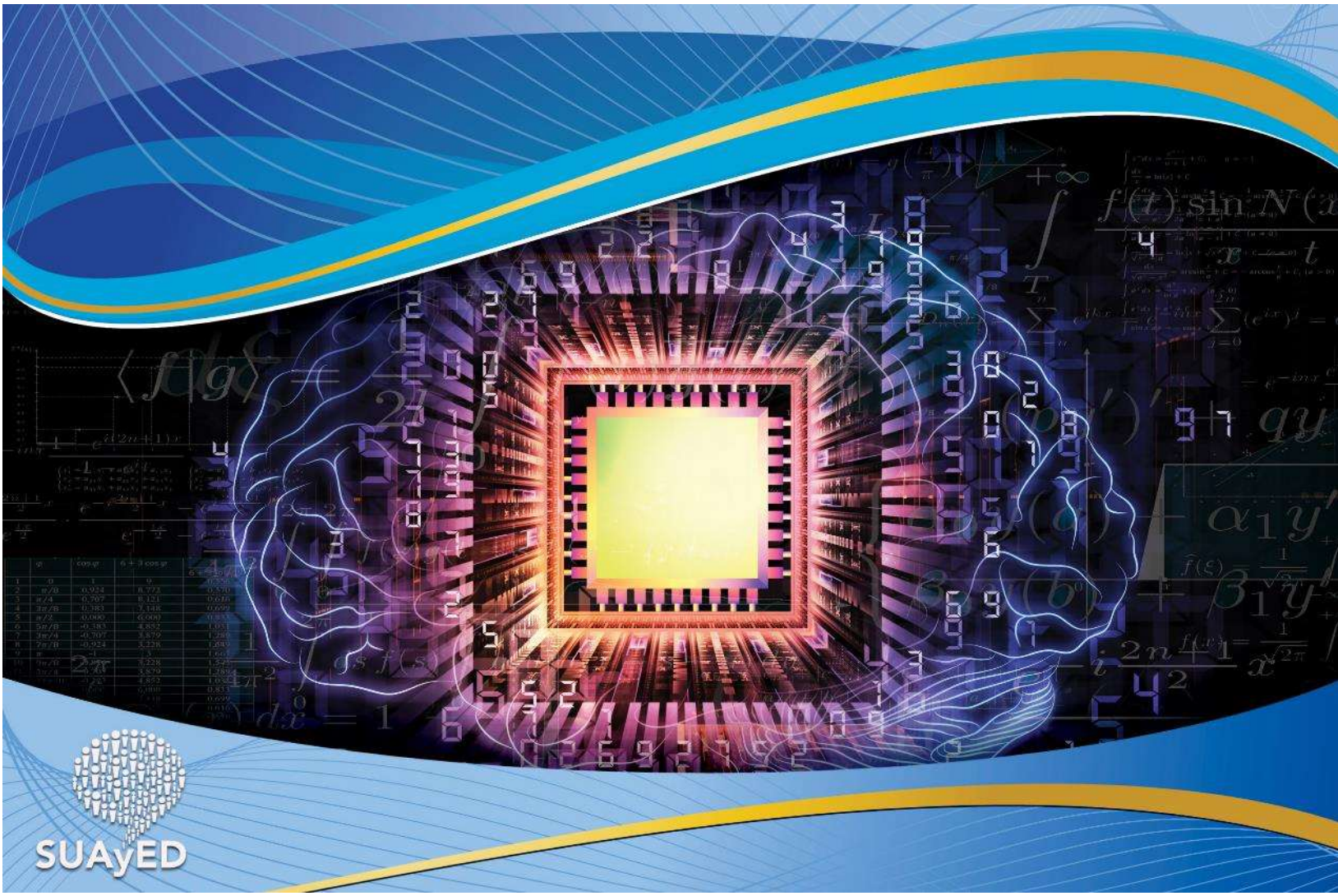
Braude, Eric J. (2003). *Ingeniería de Software: una perspectiva orientada a Objetos*. México: Alfaomega.

Kendall, Kenneth E. y Kendall, Julie E. (2005). *Análisis y diseño de sistemas*. (6ª ed.) México: Pearson Educación.

Piattini Velthuis, Mario G., *et al*. (2000). *Análisis y diseño detallado de Aplicaciones Informáticas de Gestión*. México: Alfaomega/Ra-Ma.

Unidad 3

Diseño de sistemas



OBJETIVO PARTICULAR

Al término de la unidad, el alumno podrá identificar los conceptos y principios del *diseño estructurado* para modelar y diseñar sistemas.

TEMARIO DETALLADO

(16 horas)

3. Diseño de sistemas

- 3.1. Principios del diseño estructurado
- 3.2. Conceptos del diseño estructurado
  - 3.2.1. Abstracción
  - 3.2.2. Refinamiento
  - 3.2.3. Modularidad
  - 3.2.4. Independencia funcional
  - 3.2.5. Cohesión
  - 3.2.6. Acoplamiento
  - 3.2.7. Arquitectura de software
  - 3.2.8. Jerarquía de control
  - 3.2.9. División estructural
  - 3.2.10. Estructura de datos
  - 3.2.11. Procedimiento
  - 3.2.12. Ocultamiento de información

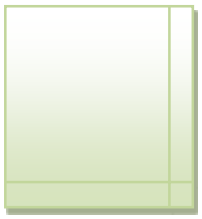
- 3.2.13. Concurrencia
- 3.2.14. Verificación
- 3.3. Diseño arquitectónico
  - 3.3.1. Arquitectura de software
  - 3.3.2. Organización del sistema
  - 3.3.3. Descomposición orientada a flujos de funciones
- 3.4. Modelado de procesos
  - 3.4.1. Diagramas HIPO
  - 3.4.2. Diagramas Nassi-Schneiderman
  - 3.4.3. Diagramas Warnier/Orr
- 3.5. Diseño de la interfaz de usuario
  - 3.5.1. Tipos de interfaces de usuario
  - 3.5.2. Reglas de oro para el diseño de interfaces de usuario
  - 3.5.3. Criterios para el diseño de interfaces de usuario
    - 3.5.3.1. Consistencia
    - 3.5.3.2. Corrección de errores
    - 3.5.3.3. Metáforas
    - 3.5.3.4. Ergonomía y estética
    - 3.5.3.5. Interfaces dinámicas
  - 3.5.4. Modelos de diseño de interfaces de usuario
  - 3.5.5. Problemas de diseño de interfaces de usuario
  - 3.5.6. Herramientas para la implementación de interfaces de usuario
  - 3.5.7. Evaluación de las interfaces de usuario

3.1. Principios del análisis estructurado

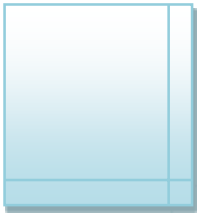
Los principios de diseño estructurado son los siguientes:

- Descomposición por refinamientos sucesivos
- Creación de una jerarquía modular
- Elaboración de módulos independientes

Módulo



Módulo Predefinido



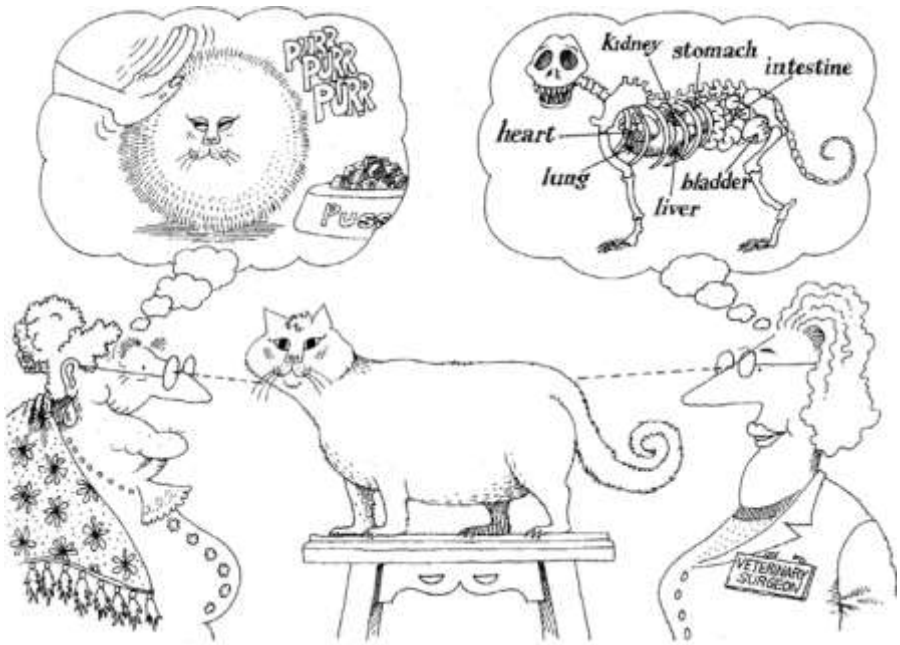
Conector

3.2. Conceptos del diseño estructurado

El diseño de sistemas se caracteriza por hacer uso de un conjunto de conceptos que deben ser comprendidos con el fin de poder especificar una propuesta de solución exitosa.

3.2.1. Abstracción

Es el proceso por el cual se resaltan ciertas características y se descartan o minimizan otras características de la realidad. Para el diseño, este proceso está enfocado en construir la solución a diferencia del análisis, el cual busca especificar la situación actual.



Abstracción (Booch, 2007; 45)

El concepto de abstracción tiene relevancia en el sentido de que los intereses, gustos, conocimientos, etc., afectan la forma en que percibimos la realidad.

3.2.2. Refinamiento

Es el proceso por el cual se refinan los procesos por automatizar en su mínima expresión, que son las actividades. Como en el diseño la materia prima son los algoritmos, las actividades deben ser transformadas en algoritmos que se deben especificar de acuerdo con el lenguaje de programación seleccionado, debido a las singularidades que tiene cada lenguaje.

No solamente se debe refinar los algoritmos, también se tiene que refinar los datos. Aunque esta tarea se hace en el análisis, cuando se crea el Diccionario de Datos, se presenta la misma situación que con los algoritmos: es necesario refinarlos en función del Sistema Manejador de Datos seleccionado.

3.2.3. Modularidad

Es el proceso por el que se agrupan o se dividen los componentes de un sistema que, a su vez, están formados por un conjunto de programas. La forma de hacer la agrupación está en función de la arquitectura que se haya seleccionado, así como de los estándares que maneje el equipo de desarrollo y la organización a la que se le está construyendo el sistema.
La ventaja de la modularización es que permite mantener y reutilizar los componentes del sistema, siempre y cuando se haya tenido una alta cohesión y un bajo acoplamiento (como se mencionó en la unidad 1).

Modularidad (Booch, 2007, p. 54)

En el libro *Análisis y Diseño orientado a Objetos con Aplicaciones*, de Booch (2007), se ilustra el concepto de modularidad con la anterior figura: observa que cada componente que conforma el gato electrónico puede ser sustituido por otro en caso de que se descomponga.

3.2.4. Independencia funcional

La independencia funcional es el resultado de aplicar la modularidad en el diseño de sistema; sin embargo, esta afirmación puede resultar un tanto falsa, ya que cada elemento del sistema se encuentra interrelacionado con, por lo menos, otro elemento. Cuando se da un cambio en el comportamiento de un elemento, éste afectará en cierto grado a otro elemento, por lo que la independencia tiende a ser virtual.

3.2.5. Cohesión

La *cohesión* es la especificidad de la función que realiza un elemento. Pressman (2001) lo define como:

> Una extensión natural del concepto de ocultación de información… Un módulo cohesivo lleva a cabo una sola tarea dentro de un procedimiento de software, lo cual requiere poca interacción con los procedimientos que se llevan a cabo en otras partes de un programa. Dicho de otra manera sencilla, un módulo cohesivo deberá (idealmente) hacer una sola cosa. (p. 231)

3.2.6. Acoplamiento

El acoplamiento es el nivel de interdependencia que existe entre los elementos. Pressman (2001) lo define como:

> Una medida de interconexión entre módulos dentro de una estructura de software. El acoplamiento depende de la complejidad de interconexión entre los módulos, el punto donde se realiza una entrada o referencia a un módulo, y los datos que pasan a través de una interfaz. (p. 231)

3.2.7. Arquitectura del software

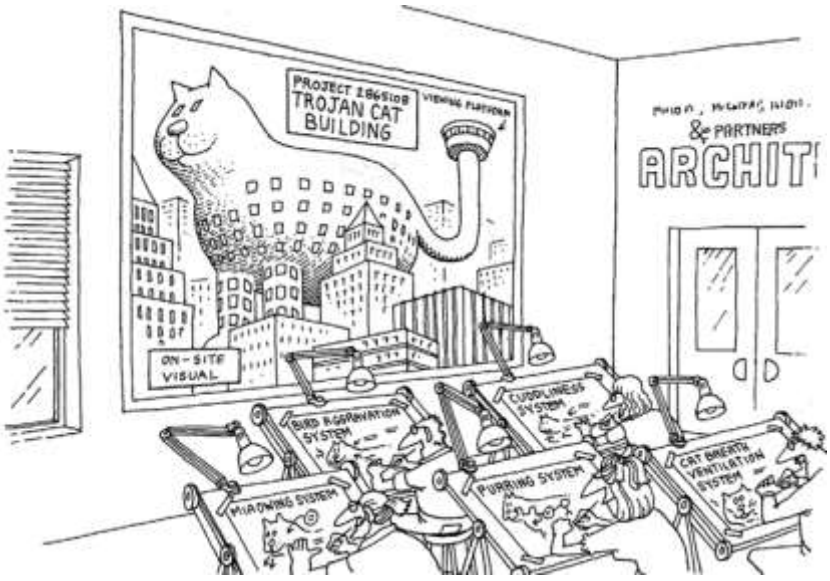
La arquitectura es “la estructura jerárquica de los componentes del programa (módulos), la manera en que los componentes interactúan y la estructura de datos que van a utilizar los componentes” (Pressman, 2001, p. 226). El diseñador establece la arquitectura, obedeciendo las características que el sistema debe tener, y la tecnología que se va a emplear, por lo que no es de extrañarse que exista una diversidad de formas para crear la arquitectura de un sistema.

Otra forma de plantear la arquitectura de software es creando un conjunto de modelos que representen la parte estática y la parte dinámica del sistema. Los modelos se representan por medio de diagramas junto con sus especificaciones correspondientes.

3.2.8. Jerarquía de control

La jerarquía representa la organización de los componentes del sistema, instaurando un escalafón en donde se establece cuál componente controla a cuál. Para este concepto es necesario establecer una regla: un componente sólo puede ser controlado por otro componente.

La jerarquía no es otra cosa más que la forma en la que se agrupan los componentes. La agrupación obedece a las características que comparten cada uno de ellos, buscando una administración eficiente. No hay un límite para establecer cuántos componentes puede controlar un componente pero se recomienda que el número no sea mayor de siete.



Jerarquía de Control (Booch, 2007)

Observa en la figura cómo el grupo de diseñadores está especificando cada una de las partes del gato electrónico; no es al azar que se encuentren en un mismo cuarto, ya que, debido a que los componentes se encuentran interrelacionados entre sí es necesario que el equipo de trabajo tenga una comunicación directa con la finalidad de compartir las interfaces que hay en los componentes.

3.2.9. División estructural

La división estructural es un concepto inherente al concepto de jerarquía de control. Como se mencionó anteriormente, es en la jerarquía de control como se organizan los componentes agrupándose entre sí. Estas agrupaciones se dan en forma horizontal (ancho) y en forma vertical (profundidad).

3.2.10. Estructura de datos

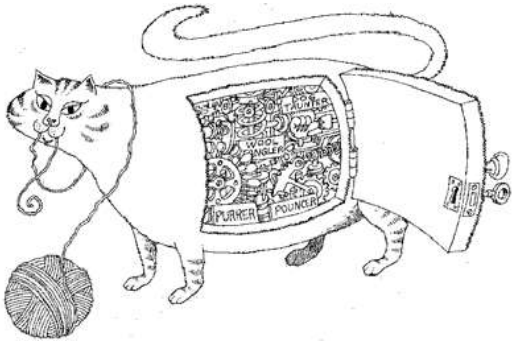
La estructura de datos es una relación entre datos. Conforme con los requerimientos del usuario, éste establecerá ciertos conceptos que tienden a convertirse en estructuras de datos. Por ejemplo, el usuario menciona que “renta películas”, la renta es una estructura de datos que está compuesta por una fecha de renta, fecha de devolución, películas rentadas, cliente que renta, entre otros.

3.2.11. Procedimiento

El procedimiento es la organización mínima de un conjunto de actividades. Se especifica por uno o más algoritmos, dependiendo de la complejidad que tenga el procedimiento. Cada procedimiento debe definir una secuencia lógica de las actividades así como los diversos caminos que pueden tomar las actividades.

3.2.12. Ocultamiento de información

Cuando se crea un componente es necesario que posea una cohesión alta, esto quiere decir que sólo debe llevar a cabo una tarea en la medida de lo posible. La tarea requiere de manipular datos para llevar a cabo su cometido, por lo que el componente tiene la responsabilidad de que estos datos sólo sean visibles para su comportamiento. Esta característica es lo que se conoce como “ocultamiento de la información”.

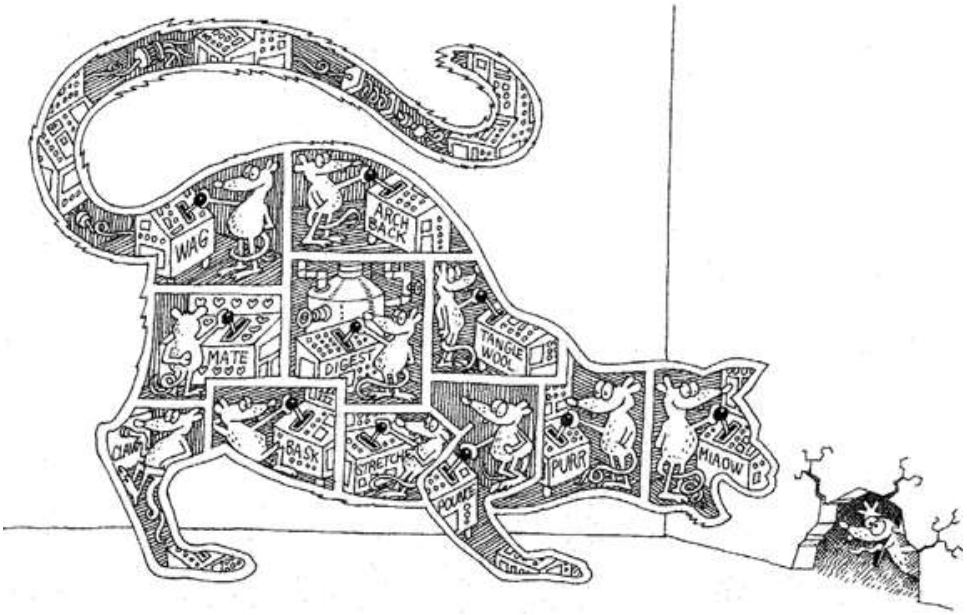


Ocultamiento (Booch, 2007)

El ocultamiento de información es una característica muy apreciada por los usuarios y desarrolladores. A los usuarios no les interesan los detalles técnicos de los sistemas, pero sí les importa que den la respuesta esperada. A los desarrolladores les interesa ocultar los detalles técnicos de los componentes con el fin de que se facilite su reutilización.

3.2.13. Concurrencia

Esta propiedad del software se da cuando tiene que responder a las peticiones de los usuarios al mismo tiempo. En entornos multiusuario tiene un peso mayor, por lo que será necesario establecer pruebas de rendimiento que permitan validar esta característica.



Concurrencia (Booch, 2007, p. 67)

3.2.14. Verificación

Es el proceso que revisa que el software se construya de la manera correcta. Al iniciar el desarrollo de un sistema es necesario establecer un conjunto de estándares para los documentos, algoritmos, bases de datos y cualquier otro tipo de artefacto. La verificación se encarga de asegurar que, conforme con los estándares definidos, los productos los cumplan; este proceso busca asegurar la calidad interna del software.

3.3. Diseño Arquitectónico

El diseño arquitectónico busca especificar una solución con base en el paradigma que se esté utilizando, en este caso, el estructurado. Su objetivo es definir la estructura de los programas y los datos que conformarán el software.

En este sentido, la estructura define la organización de cada uno de los componentes con el objetivo de que respondan a los requerimientos de los usuarios. Esta condición amerita que antes de construir los programas y los datos, primero se tienen que especificar los algoritmos, la base de datos y cómo se organizarán y comunicarán los componentes.

3.3.1. Arquitectura de software

La arquitectura de software establece la organización de los componentes para constituir el sistema que se está solicitando. Al igual que un arquitecto, el diseñador tiene que modelar diferentes aspectos que conforman un sistema. El arquitecto no es el responsable de instalar la tubería de un edificio o de instalar la electricidad de una casa, pero sí es responsable de crear los modelos para un plomero y un electricista que sí lo harán.

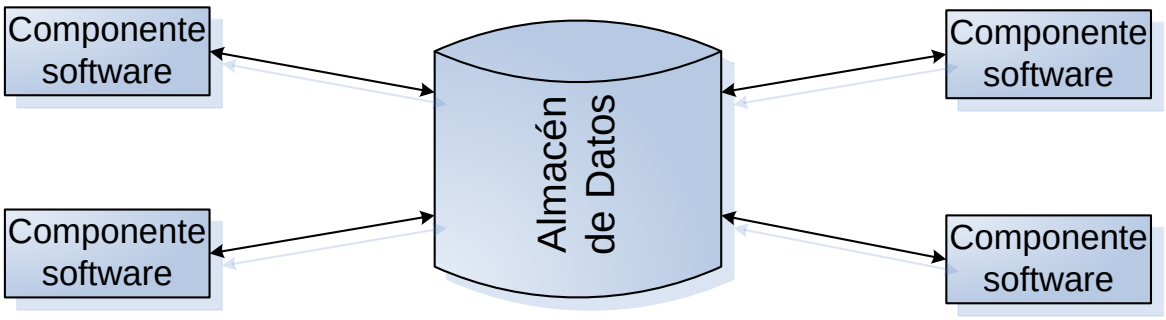
Con el ejemplo anterior queda claro que el diseñador construirá una diversidad de modelos con el fin de que programadores, diseñadores gráficos y administradores de bases de datos puedan realizar sus actividades para la construcción del sistema. Al igual que el arquitecto, el diseñador tendrá que generar diferentes versiones de los modelos a fin de satisfacer los requerimientos de los usuarios y clientes.

Estilos arquitectónicos

Un estilo arquitectónico establece un marco de referencia para crear el diseño. Como tal, establece una serie de reglas y recomendaciones que se deben seguir para establecer la organización de los componentes del sistema. El estilo arquitectónico no es una receta de cocina que deba seguirse al pie de la letra, por la razón de que la variable “persona” crea un alto nivel de volatilidad en la construcción del sistema; en consecuencia, se recomienda que el estilo que se elija sea flexible y, a la vez, robusto.

Arquitectura centrada en los datos

En esta arquitectura, como su nombre lo indica, las decisiones de diseño están orientadas a la centralización de los datos. En este estilo, el software accede a un almacén centralizado de los datos para agregar, eliminar, modificar y/o recuperar alguno de los datos contenidos en él. La ventaja de este modelo consiste en la independencia de los datos, es decir, el software debe estar construido de tal manera que si uno de sus componentes es sustituido no se verá afectado el almacén de datos.

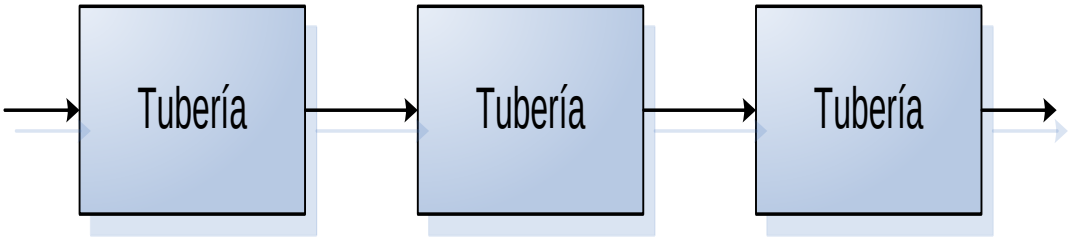


Arquitectura centrada en los datos

Arquitectura de flujo de datos

Esta arquitectura se centra en la transformación de los datos de entrada para obtener los datos de salida. Para lograr esto se utiliza los componentes del software como tuberías, estas tuberías están interconectadas entre sí trabajando en forma

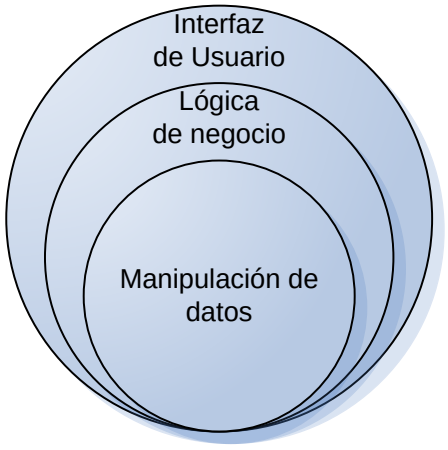
independiente. Cada tubería solo está consciente de que debe recibir los datos de cierta manera y generar una salida para la siguiente tubería, esto significa que cada tubería desconoce la forma en que trabajan las demás, sólo conoce que debe recibir la otra tubería si es que están interconectadas.



Arquitectura de flujo de datos

Arquitectura estratificada

En esta arquitectura se crea una cantidad definida de capas, cada capa tiene una responsabilidad claramente definida. Las capas externas se orientan al uso del sistema y las capas internas se orientan a la manipulación de la computadora. Dependiendo de cómo se aplique el estilo puede tener dos a más capas, de manera tradicional se utilizan tres capas: la primera tiene como responsabilidad la presentación de los datos al usuario, la segunda tiene como responsabilidad ejecutar la lógica de negocio, y la tercera tiene como responsabilidad la manipulación de los datos.



Arquitectura estratificada

Es necesario recalcar que estos estilos en la realidad son meramente teóricos y que más bien son utilizados en forma combinada, dada las condiciones cambiantes en las que se desenvuelve el desarrollo de sistemas. Incluso, el diseñador se puede encontrar en la situación de que ninguno de ellos le ayude a resolver problema y tenga que elaborar un estilo propio.

3.3.2. Organización del sistema

- Refleja la estrategia básica que es usada para estructurar un sistema.
- Tres estilos organizativos muy utilizados:
    - Modelos de repositorio.
    - Modelo cliente-servidor.
    - Modelo de capas.

Modelo de repositorio

Es comúnmente usado para grandes volúmenes de datos, en dos formas:

- Bases de datos compartida
- Bases de datos distribuidas

Para los subsistemas el acceso a los datos es transparente

Modelo cliente servidor

En este modelo los datos y procedimientos están en servidores que proporcionan servicios como impresión, compartir espacio de disco, aplicaciones remotas, etc.

La otra parte del modelo son los clientes que solicitan esos servicios a través de una red de datos donde también está el servidor.

Modelo de capas

Se usa en sistemas con varios módulos o subsistemas para modelar las interfaces entre ellos. En cada capa se realiza un proceso o servicio de tal forma que si hay un cambio en la interfaz o servicio este solo afecta a la capa pero puede ser complicado estructurar sistemas con este modelo.

Un buen ejemplo es el modelo OSI para redes de datos donde cada capa realiza una tarea específica. Si iniciáramos de la capa 7 (aplicación) en ella sólo participan las aplicaciones encargados de la interfaz con el usuario, por ejemplo internet Explorer, Firefox, Messenger, Outlook, etc. Si llegase a fallar internet Explorer se puede usar otra interfaz como Chrome que traducirá las peticiones del usuario a las demás capas para que usuario pueda usar el servicio de web.

3.3.3. Descomposición orientada a flujos de funciones

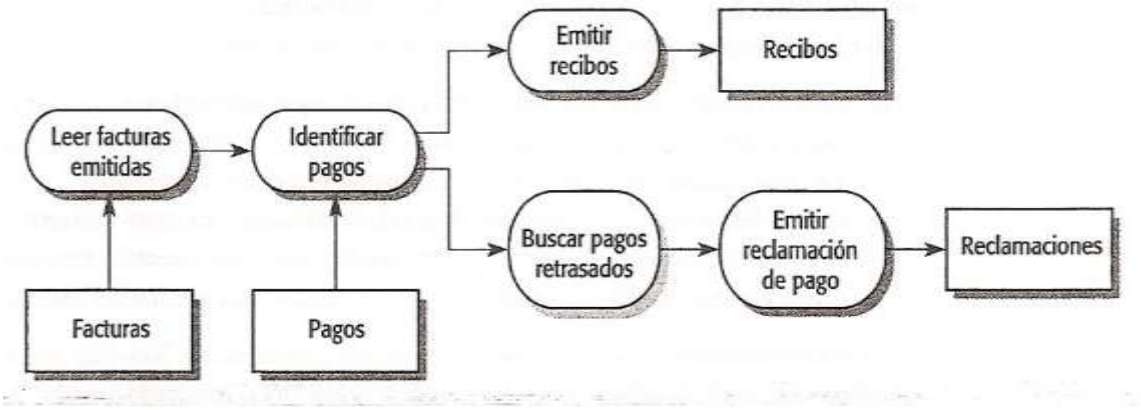
Cuando los problemas son sencillos es razonable pensar en los pasos que deberíamos dar para resolverlo y crear programas a partir de cada uno de esos pasos, pero puede no ser la mejor forma de actuar cuando se trata de algo más complicado. En esos casos lo mejor es descomponer los problemas en trozos más pequeños que sean más fáciles de resolver, evocando la famosa frase de “divide y vencerás”. Esto nos puede suponer varias ventajas:

- Cada parte del programa será más fácil de programar, al realizar una función breve y concreta.
- El programa principal será más fácil de leer, porque no necesitará contener todos los detalles de cómo se realiza cada proceso.
- Podremos repartir las labores de programación a un grupo de trabajo donde cada integrante se encargue de realizar una parte del programa, y finalmente se integrará el trabajo individual de cada persona. Esos "trozos" de programa son conocidos como subrutinas o funciones.

La descomposición orientada a flujos de funciones consiste en separar los problemas complejos con base en lo que realizan (función) para facilitar la lectura de los programas y reutilizar el código.

Ventajas

- Permite reutilización de transformación.
- Organización intuitiva para comunicación del grupo de trabajo.
- Fácil añadir nuevas transformaciones
- Relativamente simple de poner en práctica como un sistema concurrente o como secuencial.



Ejemplo de flujo de funciones para una facturación

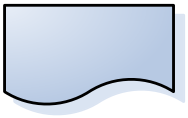
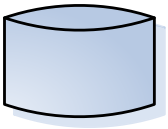
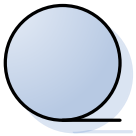
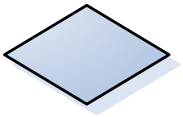
3.4. Modelado de Procesos

La forma más común de modelar procesos en el análisis y diseño estructurado son los Diagramas de Flujos de Datos (DFD); sin embargo, en ocasiones es necesario apoyarse de otras herramientas como los diagramas

- HIPO
- Nassi-Schneiderman
- Warnier/Orr

3.4.1. Diagramas HIPO

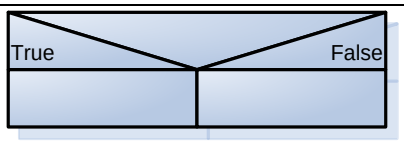
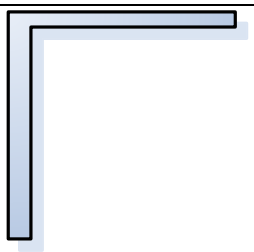
El acrónimo HIPO corresponde a *Hierarchy-Input-Process-Output*, que podría traducirse como “Jerarquía de entrada-proceso-salida”. Los diagramas HIPO buscan especificar un proceso con base en su entrada o el proceso para procesar las entradas en la salida y la salida. Los símbolos que se utilizan en este tipo de diagramas son los siguientes:

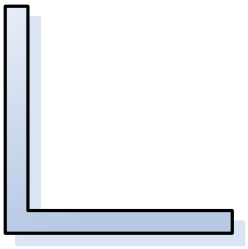
Símbolo	Descripción
	Programa. Es cualquier actividad llevada a cabo por el sistema.
	Listado. Es un conjunto de registros del mismo tipo.
	Utilidad o rutina. Es una actividad en su mínima expresión, es decir, ya no se puede fragmentar en otras actividades.
	Disco. Representa un medio de almacenamiento en donde se guardan datos para su posterior recuperación.
	Cinta magnética. Representa un medio de almacenamiento electromagnético.
	Utilidad *sort*. Representa una actividad de ordenamiento.

Símbolos del Diagrama HIPO

3.4.2. Diagramas Nassi-Schneiderman

Los diagramas Nassi-Schneiderman son similares a un diagrama de flujo pero omite las flechas que representan el flujo de control. En lugar de las flechas se utilizan rectángulos colocados en forma sucesiva; dentro de cada rectángulo se escribe la instrucción en forma de pseudocódigo u otro conjunto de rectángulos o de símbolos para representar una condición o ciclo. Los símbolos que se utilizan en este tipo de diagramas son:

Símbolo	Descripción
	Acción. Con este símbolo se representa una instrucción, operación u expresión del algoritmo. El rectángulo puede ponerse dentro de otros símbolos para representar una condición o ciclo.
	Condición. Con este símbolo se representa una condición lógica. Como se puede observar, en el lado izquierdo está la palabra *true* y debajo de ella un rectángulo donde se ponen las instrucciones cuando la condición se cumple, si es necesario se pueden poner más rectángulos. Del lado derecho está la palabra *false* y debajo de ella un rectángulo para poner las instrucciones cuando la condición no se cumple.
	Ciclo mientras. Con este símbolo se representa el ciclo mientras, lo cual quiere decir que todos los rectángulos que estén contenidos en él se repetirán mientras la condición se cumpla. Si la condición es falsa desde un principio, los rectángulos contenidos no se ejecutarán.

	Ciclo repetir. Con este símbolo se representa el ciclo repetir, lo cual quiere decir que todos los rectángulos que estén contenidos en él se repetirán mientras la condición se cumpla. En este ciclo, por lo menos se ejecutan una vez los rectángulos contenidos en él.

Símbolos del Diagrama Nassi-Schneiderman

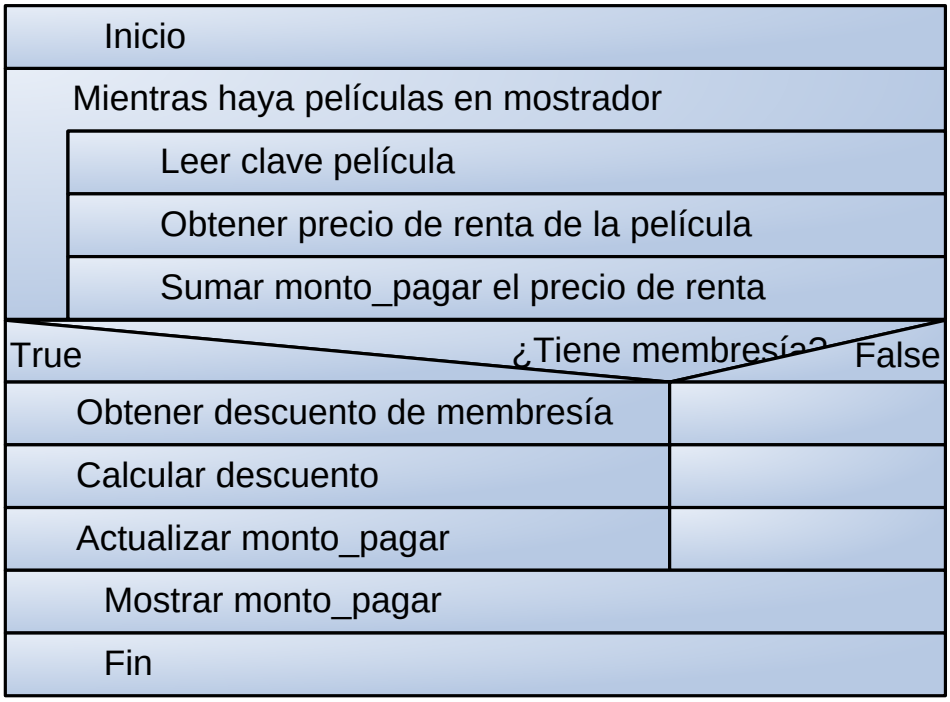


Diagrama Nassi-Schneiderman del cobro de una renta de películas

Continuando con el ejemplo de la renta de películas, si se necesitara especificar el algoritmo para detallar el cobro de renta de películas, el ejemplo de la figura anterior sería una posible solución para representar ese algoritmo.

3.4.3. Diagramas Warnier/Orr

Este diagrama permite la descripción de los procedimientos y los datos de la organización y fueron desarrollados inicialmente en Francia, por Jean-Dominique Warnier y Kenneth Orr. Estos diagramas permiten diseñar un programa identificando primero su salida, para trabajar hacia atrás y poder determinar las combinaciones de pasos y entradas para producirlas. La ventaja de estos diagramas consiste en su apariencia simple y sencilla de entender. En ellos se agrupan los procesos y datos de un nivel a otro.

Para hacer estos diagramas, el diseñador trabaja de atrás hacia adelante, comienza con las salidas del sistema y en un papel escribe de derecha a izquierda. Primero se ponen salidas o resultados de los procesos. En el siguiente nivel se escriben en una inclusión, con una llave, los pasos necesarios para producir la salida. Finalmente, las llaves agrupan los procesos requeridos para producir el resultado del siguiente nivel, sus elementos son los siguientes:

Elemento	Descripción
{ **Conjunto**	Delimita un bloque de información jerarquizada (datos o acciones); de derecha a izquierda denota los niveles de abstracción, de arriba abajo muestra la secuencia y las relaciones lógicas entre las funciones.
(0,1) **Condicionalidad**	La información entre los paréntesis (variable o cantidad) indica el número de veces que ocurrirá el conjunto. Si se coloca una letra “C” indica que un ciclo se termina cuando la condición se cumpla.
+ **Secuencia de acciones mutuamente excluyentes**	Indica que una acción o grupo de acciones son mutuamente excluyentes, dadas las condiciones que se establezcan.

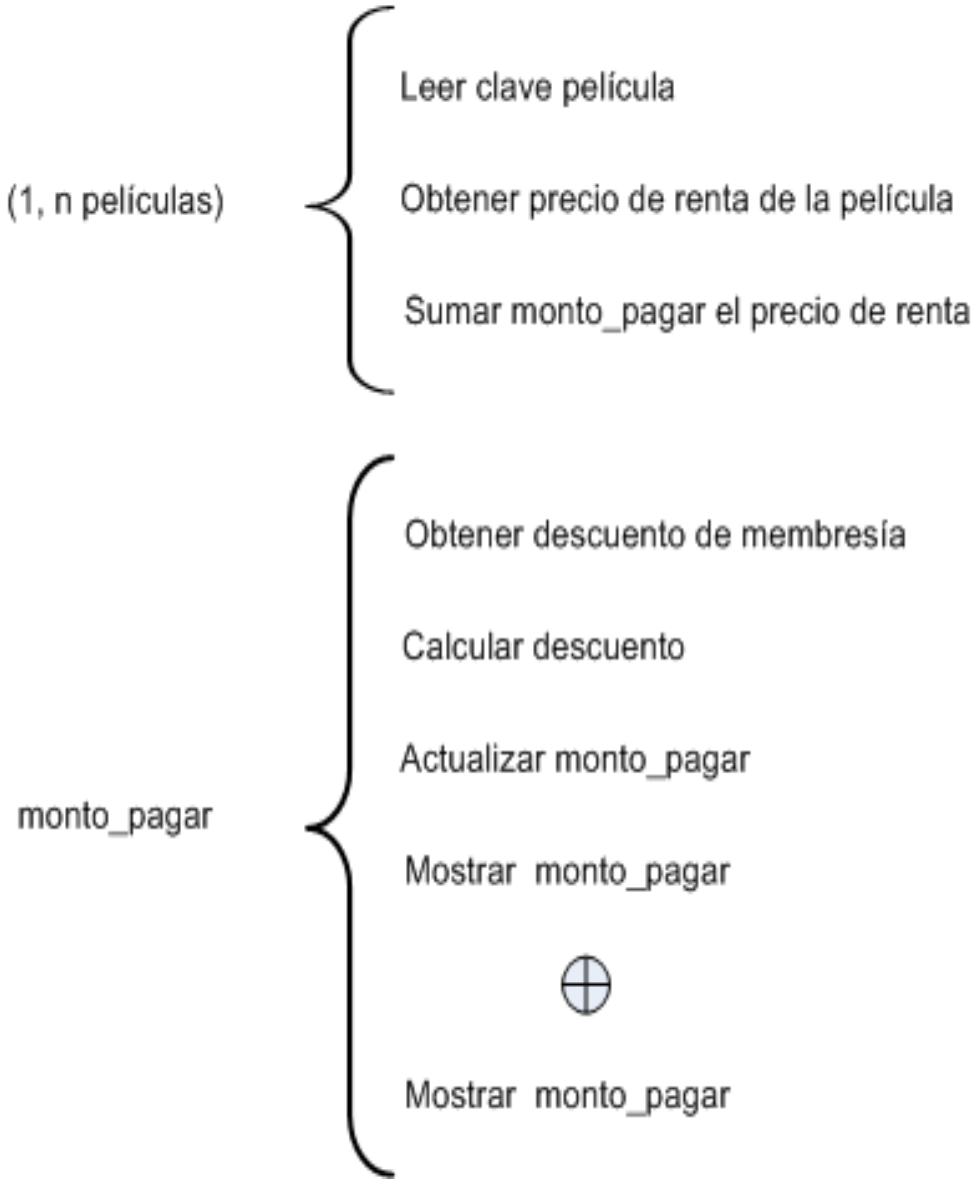


Diagrama Warnier/Orr del cobro de una renta de películas

3.5. Diseño de la interfaz de usuario

Uno de los elementos en los que el diseñador debe tener sumo cuidado es en la interfaz de usuario. La percepción que tenga el usuario en cuanto a la facilidad de uso impactará en su percepción de utilidad que tenga del sistema. El diseñador debe conocer cuáles son los conocimientos que tiene el usuario con respecto al uso de software, esto le da un marco de referencia para construir una interfaz que se adecue a las necesidades y costumbres que tenga el usuario.

El diseño de la interfaz de usuario es una actividad que se realiza con el apoyo de diseñadores gráficos por tener conocimientos más especializados en cuanto a la ergonomía, navegabilidad, colores, brillo, nitidez. Sin embargo, no siempre se cuenta con este apoyo por lo que se recomienda profundizar en estos conocimientos y establecer una comunicación continua con el usuario respecto a las decisiones por tomar con la interfaz.

3.5.1. Tipos de interfaces de usuario

Las interfaces de usuario se pueden clasificar según la interacción que hay entre el usuario y el sistema, los cuales son:

- Línea de comando
- Menú de selección
- Llenado de forma
- Manipulación directa
- Antropomórfica

La selección del estilo de interacción está delimitada por el tipo de sistema por desarrollar y de las características de los dispositivos de entradas y salida que se utilizarán en la interfaz. A continuación se describe cada uno de los tipos de interacción:

Línea de comando

Es una de las más viejas en la que se requiere que el usuario presione una tecla de función o teclee un comando en un área de la pantalla. Los comandos pueden ser letras, abreviaturas, palabras, o múltiples palabras y funciones. La línea de comandos es un estilo poderoso que ofrece acceso inmediato a funciones del sistema. Además, tiene la flexibilidad y facilidad para incorporar opciones o parámetros. El problema con la línea de comandos es que no siempre es fácil recordar los comandos, más si se encuentran en otro idioma o abreviaturas desconocidas.

Menú de selección

Un menú es un conjunto de opciones que el usuario puede elegir. En las pantallas, el usuario selecciona una opción con un puntero o presionando una combinación de teclas. La mayoría de las veces, algunos menús retroalimentan al usuario indicando la opción seleccionada. Los menús tienen la ventaja de que los usuarios reconocen las opciones sin necesidad de memorizar comandos con todas sus opciones. Sin embargo, los nombres de las opciones están limitados por el espacio.

Llenado de forma

Este estilo es muy usado para recolectar información. Las formas están estructuradas en pantallas que contienen controles o campos en que el usuario proporciona datos o selecciona una opción u opciones. Los antecedentes del

llenado de forma están en las formas de papel por lo que trae como consecuencia su familiaridad por parte de los usuarios.

Manipulación directa

Esta se encuentra en los sistemas gráficos, permite al usuario interactuar directamente con los elementos que se presentan en las pantallas. Estos elementos reemplazan la acción de teclear comando o seleccionar menús. Los usuarios seleccionan los objetos de la pantalla y las acciones usando el ratón o joystick. La navegación en la pantalla y la ejecución de los cambios es accediendo a una barra de menú con opciones desplegables.

Antropomórfica

Una interfaz antropomórfica intenta interactuar con la gente tal como la gente interactúa entre sí. Para ello es necesario que la interfaz pueda establecer un diálogo en lenguaje natural, gestos manuales, expresiones faciales, y movimiento de ojo. El desarrollo de estas interfaces requiere un entendimiento del comportamiento humano; como la gente interactúa una con otra, el significado de los gestos y expresiones, etc.

Estos estilos no son mutuamente excluyentes, es decir, se pueden combinar según se necesite así como de la tecnología que se vaya a ocupar para el desarrollo del sistema.

3.5.2. Reglas de oro para el diseño de interfaces de usuario

Theo Mandel (1997) estableció tres “reglas de oro” para el diseño de la interfaz:

- **Dar el control al usuario**. Esta regla busca crear una interacción con el usuario donde el usuario no esté obligado a ejecutar acciones no necesarias y/o no deseadas. La interacción debe ser flexible, es decir, llegar a una funcionalidad desde diferentes formas o caminos. Permitir al usuario interrumpir las acciones que ejecuta con el sistema en cualquier momento. Permitir al usuario modificar la forma en que interactúa con el sistema. Ocultar errores, advertencias o cualquier otro elemento técnico que corresponda a la implementación del sistema.
- **Reducir la carga de memoria del usuario**. Esta regla busca minimizar el esfuerzo por parte del usuario para tener que recordar cómo tiene que interactuar con el sistema. La regla está relacionada con la petición “una interfaz amigable” que expresan los usuarios en las reuniones. Para ello se debe tratar de diseñar elementos que sean intuitivos, diseñar elementos lo más aproximado posible a los objetos de la vida real, proporcionar información adicional de cada elemento cuando se requiera.
- **Construir una interfaz consistente**. Esta regla busca la consistencia de la información en cada una de las pantallas, las acciones de cada uno de los elementos sean consistentes evitando sorpresas para el usuario y mantener la secuencia de acciones coherente.

3.5.3. Criterios para el diseño de interfaces de usuario

Lo que se busca con las interfaces es que sean una extensión de las personas, es decir, el sistema debe ser acorde con las capacidades de las personas y que responda en forma específica a sus necesidades. Esto implica que la interfaz busca apoyar en la consecución de los objetivos de la organización en forma eficiente, esto lo logra si es fácil de aprender y fácil de usar evitando sentimientos de tedio o frustración. Para lograr eso se ha generado un conjunto de criterios mínimos que deben ser tomados en cuenta al momento de crear una interfaz que se explican a continuación:

3.5.3.1. Consistencia

La consistencia gravita en la uniformidad en apariencia, colocación, y comportamiento. Esta es una regla que orienta a todas las actividades de diseño. La importancia de este criterio está en que puede reducir el esfuerzo humano para aprender, así como las habilidades requeridas para su aprendizaje.

3.5.3.2. Corrección de errores

El usuario debe tener la posibilidad de retractarse o cancelar una acción haciendo uso de un comando de deshacer. Esta característica ayuda mucho a los usuarios nuevos disminuyendo su estrés cuando hacen algo mal. Se debe tener mucho cuidado cuando una acción no se puede deshacer y sus consecuencias son críticas, hay que asegurar que los usuarios nunca pierdan su trabajo como resultado de sus errores o errores técnicos.

3.5.3.3. Metáforas

Muchas veces, para simplificar descripciones que ejecutan cierta tarea se sustituyen por un elemento visual que representa la acción que se espera que haga el usuario, a esto se le llama metáforas. Las metáforas o analogías deben ser realistas y simples. Además, se debe tratar de que se asemejen lo más posible a la vida real y que tengan significado para el usuario.

3.5.3.4. Ergonomía y estética

Los sistemas son más usables cuando indican claramente su estado, las posibles acciones que se pueden tomar y los resultados de las acciones que se hagan con él. Esto se logra creando una organización jerárquica colocando la información o controles dentro de categorías lógicas. También, es necesario presentar y ocultar la información y control según el contexto en el que se encuentre el sistema.

Las características anteriores se logran aplicando los conceptos de ergonomía y de estética. La ergonomía es un proceso que busca encontrar la distribución de los elementos más adecuada a fin de que las personas ejecuten sus actividades eficientemente. La estética busca que los elementos de la interfaz gráfica sean agradables a la vista del usuario. Aunque ambos conceptos pueden ser aplicados por separado, su fortaleza radica en su aplicación conjunta.

3.5.3.5. Interfaces dinámicas

Por su naturaleza, la de ser un intermediario entre dos entes (máquina-usuario), todas las interfaces gráficas son de carácter dinámico. El término de interfaces dinámicas en algunas ocasiones es sinónimo de interfaces amigables y se presenta en la fase de análisis cuando algún usuario o cliente quiere dar a entender que

espera una interfaz gráfica atractiva y que sea fácil de usar a fin de agilizar su trabajo.

3.5.4. Modelos de diseño de interfaces de usuario

Cuando se está en el proceso de desarrollo de un sistema se puede presentar uno o más de los siguientes modelos, con respecto a la interfaz de usuario:

- Modelo de diseño. Es creado por un ingeniero de software.
- Modelo de usuario. Es creado por cualquier otro ingeniero que no sea ingeniero de software.
- Percepción de usuario. Es creado por el usuario final.
- Imagen del sistema. Es creado por los programadores del sistema.

Cabe destacar que estos modelos pueden ser meramente mentales según lo cree cada persona por lo que el diseñador de la interfaz tiene como tarea conciliar estos modelos a fin de considerar los puntos de vista y el sistema cumpla con su propósito.

3.5.5. Problemas de diseño de interfaces de usuario

Los problemas que se presentan en el diseño de interfaces de usuario por lo regular caen en los rubros de: tiempo de respuesta, servicios de ayuda e información sobre errores.

Tiempo de respuesta. En cuanto el tiempo de respuesta, hay dos características que tomar en cuenta: duración y variabilidad. El tiempo que tarda el sistema en presentar la respuesta (duración) puede crear ideas equivocadas del sistema, un tiempo muy corto le podría indicar al usuario un sentido de urgencia provocándole que se precipite y cometa un error, un tiempo muy largo le podría provocar

frustración o que algo hizo mal. La variación de tiempo a un mismo evento (variabilidad) es un factor que también afecta la percepción del usuario respecto al sistema, en entornos Web donde la respuesta está influida por el ancho de banda es algo que muy pocas veces está al alcance del diseñador para controlar.

Servicio de ayuda. Hay dos formas de crear el servicio de ayuda de un sistema, una es durante la construcción del sistema y la otra se hace cuando se finaliza la construcción del sistema. Para ambos casos es necesario considerar los siguientes puntos:

- ¿Es la misma ayuda para todos los usuarios? ¿Según la responsabilidad del usuario con el sistema son los elementos de ayuda?
- ¿Cómo va acceder el usuario a la ayuda: impresa o electrónica? ¿Si la ayuda es electrónica será por medio del sistema, un recurso externo al sistema?
- ¿Cómo será estructurada la información en la ayuda? ¿Será la misma estructura para el formato impreso que el electrónico?

Información sobre errores. Un error con la leyenda “Error 34: Desbordamiento de memoria en tiempo de ejecución en el sector 1846” podrá ser muy útil para el programador. Sin embargo, para el usuario final carece de sentido y puede resultar atemorizante. Por esta razón es necesario tomar en cuenta las siguientes consideraciones en los mensajes de error:

- La redacción y vocabulario del mensaje debe ser acorde con el perfil del usuario.
- El contenido del mensaje debe tener instrucciones claras, si es el caso, del motivo del error así como las acciones viables para corregirlo.
- El contenido del mensaje debe ser claro en la consecuencia negativa del error para que el usuario pueda verificar el efecto.
- De ser posible, el mensaje debe tener un distintivo visual y/o auditivo.

3.5.6. Herramientas para la implementación de interfaces de usuario

El uso de las herramientas de implementación empieza cuando se tiene un prototipo de la interfaz de usuario que haya sido validado por los usuarios. Existen diversas herramientas que atienden a diferentes propósitos según el ambiente en que se va a desarrollar el sistema. No es lo mismo crear una interfaz de usuario para un ambiente web o un ambiente Windows o un ambiente Mac o un ambiente Linux u otro. Cuando se realice la investigación de la herramienta por ocupar se debe considerar que como mínimo facilite la creación de ventanas, menús, mensajes o cualquier otro elemento de la interfaz de usuario. En algunos casos es preferible que cuente también con las siguientes características:

- Validar los datos de entrada del usuario
- Gestionar mensajes (avisos, advertencia, error)
- Facilitar la creación de ayuda para el usuario
- Facilitar la interacción con base de datos para presentar los datos
- Permitir que el usuario pueda configurar la presentación de la interfaz

Analizando el listado anterior se encuentra que no siempre las herramientas contarán con estas características. Además, tal pareciera que se está programando parte de la lógica de negocio, por tal razón los líderes del proyecto con el fin de ahorrar tiempo utilizan un lenguaje de programación para construir la interfaz de usuario.

3.5.7. Evaluación de las interfaces de usuario

No hay mejor evaluación de interfaz de usuario que el propio usuario “haciendo uso” de ella. Esta actividad se puede hacer de manera informal o formal: en la manera *informal* el diseñador proporciona al usuario la interfaz y observa su comportamiento al usarla, durante el proceso el diseñador hace anotaciones respecto a los tiempos de uso y comportamiento que presenta el usuario; la manera *formal* implica una mayor inversión de tiempo, debido a que es necesario determinar las características que se esperan evaluar, construir el instrumento que las medirá, aplicarlo con el usuario y con los resultados obtenidos actualizar o corregir la interfaz.

Las características que se pueden evaluar en una interfaz gráfica son: el tiempo de respuesta del sistema, el tiempo de aprendizaje del usuario, la cantidad de acciones realizadas por el usuario para ejecutar una tarea, la claridad y pertinencia de las instrucciones y mensajes de error, la satisfacción por parte del usuario en ocupar la interfaz y cualquier otro aspecto que ayude a mejorar la interfaz.

RESUMEN

Elaborar un diseño hace referencia a la especificación del sistema. De un diseño adecuado al sistema dependerá el dar respuesta a los problemas y necesidades detectados durante el análisis. Recordemos que en el diseño, se toman las decisiones concernientes a la organización de los componentes y a la estructura de cada uno de ellos. Existen diferentes modelos que se toman en cuenta para diseñar la arquitectura de los sistemas, así como las técnicas empleadas para especificar los componentes del sistema y obtener resultados confiables. El papel que juegan los diagramas en el diseño es el de la construcción ordenada del software, ya que permiten visualizar el intercambio de información.

BIBLIOGRAFIA

SUGERIDA

Autor	Capítulo	Páginas
Bruegge (2001)	3, 9, 10,13, 14 y 15, 28	177-300, 626
Gómez y Suárez (2007)	10,11, 25	166, 217, 264-266
Johansen (2006)	16	331

Braude J., Eric. (2003). *Ingeniería de Software: Una Perspectiva Orientada a Objetos*. México: Alfaomega.

Bruegge, Bernd. (2001). *Ingeniería de software orientada a objetos.* México: Prentice Hall.

Gómez Vieites, Álvaro y Suárez Rey, Carlos. (2007). *Sistemas de Información: Herramientas prácticas para la gestión empresarial*. (2ª ed.) México: Alfaomega.

Johansen Bertoglio, Oscar. (2006). *Introducción a la Teoría General De Sistemas*. México: Limusa.

Facultad de Contaduría y Administración
Sistema Universidad Abierta y Educación a Distancia