Autoras:
XIMENA MARCILLO E.
MYRIAN PÉREZ A.

DOXA EDITION

2023

Análisis y diseño de sistemas de información

para proyectos de desarrollo de software

Primera Edición

ITQ
WWW.ITQ.EDU.EC
INVESTIGACIÓN

ISBN: 978-9942-8921-3-3

DOXA EDITION

INSTITUTO SUPERIOR TECNOLÓGICO QUITO
Excelencia en Educación Superior

ANÁLISIS Y DISEÑO DE SISTEMAS DE INFORMACIÓN PARA PROYECTOS DE DESARROLLO DE SOFTWARE

AUTORAS:

XIMENA MARCILLO E.

MYRIAN PÉREZ A.

PRIMERA EDICIÓN

2023

TRABAJO EN EDICIÓN:

DOXA EDITION

EDICIÓN INTERNA: DIEGO ORTEGA G.

EDICIÓN EXTERNA: DIEGO BASTIDAS L.

ISBN: 978-9942-8921-3-3

QUITO - ECUADOR

DOXA EDITION

DEDICATORIA

Dedicamos este libro a aquellos que han sido parte de nuestro camino y han contribuido a nuestro crecimiento personal y profesional. A nuestras familias, por su amor incondicional, apoyo y paciencia durante todo el proceso de crecimiento y desarrollo personal a través de nuevas vivencias. A nuestros amigos, por su amistad y por brindarnos momentos de alegría y distracción en los momentos más difíciles.

A todos ustedes, les agradecemos por ser nuestra motivación y por ser parte de nuestra historia. Además, dedicamos este libro a nuestros lectores, por su interés en nuestro trabajo y por darnos la oportunidad de compartir nuestra pasión por el mundo de la enseñanza. Esperamos que estas páginas les brinden conocimientos, inspiración y entretenimiento, y que les acompañen en su propio camino de crecimiento personal y profesional.

Las autoras

AGRADECIMIENTO

Queremos expresar nuestro más sincero agradecimiento a todas aquellas personas que hicieron posible la creación de este libro. En primer lugar, agradecemos a nuestras familias por su amor, apoyo y paciencia durante todo el proceso de escritura. Gracias por brindarnos el espacio y el tiempo necesario para concentrarnos en esta tarea y por ser nuestra fuente de inspiración en cada página.

También agradecemos a nuestros amigos y colegas por su constante motivación, por brindarnos retroalimentación y sugerencias valiosas durante la escritura de este libro. Gracias por su amistad y por haber compartido con nosotras momentos de alegría y distracción.

Un agradecimiento especial a nuestro editor por su profesionalismo, dedicación y paciencia en la revisión y edición de este libro. Gracias por habernos guiado en cada paso del proceso de publicación y por haber hecho posible la materialización de este proyecto.

Finalmente, queremos expresar nuestro agradecimiento a los lectores por su interés en este trabajo y por brindarnos la oportunidad de compartir nuestras ideas y pensamientos sobre los sistemas de información. Esperamos que este libro les brinde una experiencia única y significativa.

Ximena y Myrian

SOBRE LAS AUTORAS

Ximena Marcillo Espinoza es una educadora y profesional ecuatoriana con un título en Ingeniería de Sistemas. Realizó sus estudios en la Universidad Politécnica Salesiana y esta es su primera publicación. Además de su pasión por la investigación, Ximena es docente en el Instituto Superior Tecnológico Quito, donde enseña la Carrera de Desarrollo de Software. A lo largo de su carrera, ha colaborado en varios proyectos de educación e investigación.

Myrian Pérez Abril es una docente con más de 18 años de experiencia en educación a nivel medio y superior, con un enfoque especial en la gestión y administración educativa, así como en la enseñanza de la física. Es Ingeniera en Sistemas, diplomada en docencia virtual y actualmente está cursando una maestría en docencia superior en la Universidad Iberoamericana de México. Además, se desempeña como docente de la Carrera de Desarrollo de Software en el Instituto Superior Tecnológico Quito.

CONTENIDO

INDICE DE FIGURAS

ÍNDICE DE TABLAS

DOXA EDITION

INTRODUCCIÓN AL CONTENIDO DEL LIBRO

El libro se enfoca en el proceso de análisis de sistemas de información en una empresa. El objetivo es recorrer los diversos análisis de sistemas existentes para sugerir soluciones para mejorarlos o desarrollar un nuevo sistema. A lo largo del libro se proporciona una guía paso a paso para el proceso de análisis de sistemas de información.

En esta obra se cubre los conceptos básicos de los sistemas de información. Se analiza la importancia de los sistemas de información en las empresas modernas y su evolución a lo largo del tiempo. Además, se destaca que los sistemas de tecnología de la información (TI) son la base para la toma de decisiones eficaz en la empresa y la gestión eficaz de los procesos de negocio.

También se abordan los diferentes tipos de sistemas de información existentes, como los sistemas transaccionales, los sistemas de soporte a la toma de decisiones y los sistemas expertos. Se explican las características de cada uno de estos sistemas y se analizan sus aplicaciones en diferentes contextos empresariales.

Asimismo, el libro discute el proceso de análisis de sistemas de información y proporciona una visión general de las diferentes etapas que se deben seguir para llevar a cabo este proceso. También se describen las técnicas y herramientas que se pueden utilizar para recopilar información sobre los sistemas existentes, analizar los problemas y proponer soluciones.

Además, se realiza un recorrido por las diferentes metodologías de desarrollo de proyectos para crear sistemas. De igual forma, se analiza las técnicas de recopilación de datos y análisis de los diferentes tipos de documentos que hay que llevar a cabo en el desarrollo de sistemas de información. En resumen, esta obra es una guía completa para el proceso de análisis y desarrollo de sistemas de información en una empresa.

CAPÍTULO 1
FUNDAMENTOS DE ANÁLISIS DE SISTEMAS DE INFORMACIÓN

1.1. ANÁLISIS DE SISTEMAS

El análisis de sistemas es una disciplina fundamental en la gestión empresarial que se encarga de estructurar, desarrollar, mantener y evaluar los sistemas de información que respaldan las operaciones de una organización. Para ello, los analistas de sistemas se basan en el estudio de los requisitos de la empresa y combinan modelos y procesos de implementación de sistemas informáticos para satisfacer sus necesidades.

Los analistas de sistemas tienen una amplia gama de responsabilidades, incluyendo la recopilación de los requisitos de la empresa, el desarrollo de procesos y la determinación de los datos necesarios para el desarrollo del sistema. Además, también deben analizar los cambios en los requisitos de la empresa que puedan afectar el diseño del sistema. Una de sus tareas más importantes es revisar y evaluar los sistemas informáticos para garantizar que cumplan con los requisitos y objetivos de la empresa.

Otro aspecto relevante del análisis de sistemas es que permite a los desarrolladores cuantificar objetivamente los sistemas para seleccionar o actualizar la arquitectura del sistema más eficiente y obtener conocimientos técnicos. Durante el proceso de diseño, se deben realizar evaluaciones siempre que se tomen decisiones o elecciones de ingeniería para determinar el cumplimiento de los requisitos del sistema.

En definitiva, el análisis de sistemas proporciona un enfoque riguroso para la toma de decisiones, que se utiliza para realizar estudios de compensación, modelado y simulación, análisis de costos, análisis de riesgos técnicos y análisis de efectividad. Este enfoque estructurado y disciplinado es fundamental para el éxito de las organizaciones modernas, ya que permite el desarrollo de sistemas de información eficientes y una toma de decisiones más eficaz. Por lo tanto, el análisis de sistemas es una disciplina esencial en la gestión empresarial.

1.1.1 TIPOS DE ANÁLISIS

- *Análisis de Requisitos:* El análisis de requisitos es una fase crítica en el proceso de desarrollo de sistemas informáticos, ya que implica la definición y documentación exhaustiva de todos los requisitos, tanto funcionales como no funcionales, del sistema en cuestión. La recopilación de estos requisitos se realiza en estrecha colaboración con el cliente, a fin de garantizar que el sistema se

desarrolle de acuerdo a sus necesidades y expectativas. En este sentido, el análisis de requisitos no solo establece las bases para el diseño y desarrollo del sistema, sino que también ayuda a prevenir errores y fallos en etapas posteriores del proyecto.

- *Análisis de Diseño:* El análisis de diseño es un paso importante en el desarrollo de un sistema informático que incluye el diseño de la arquitectura y la lógica del sistema. Esta fase implica un análisis detallado de cada componente del sistema, incluidos los lenguajes de programación, las bases de datos, las interfaces de usuario, la seguridad, etc.
- *Pruebas de Aceptación*: La prueba de aceptación es una parte importante del proceso de desarrollo del sistema informático que implica probar el sistema frente a requisitos funcionales y no funcionales específicos. Estas pruebas[1] se realizan para garantizar que el sistema desarrollado se ajusta a los requerimientos del cliente.
- *Análisis de Rendimiento:* El análisis de rendimiento es un paso importante en el desarrollo de un sistema informático e incluye la evaluación del rendimiento del sistema. Estas pruebas se realizan para determinar la capacidad de respuesta, la estabilidad y la escalabilidad. Esta clasificación le permite mejorar el rendimiento del sistema.

1.2. SISTEMAS DE INFORMACIÓN

Los sistemas de información son fundamentales para la gestión eficiente de una empresa, ya que mejoran su productividad y rentabilidad. Constituyen una herramienta esencial que permite la gestión y administración de la información en el entorno empresarial actual.

Un sistema de información es un conjunto de procesos tecnológicos y organizativos que permiten la recopilación, procesamiento, almacenamiento y distribución de información, con el objetivo de respaldar la toma de decisiones dentro de la empresa. Estos sistemas están compuestos por diferentes elementos, tales como hardware, software, recursos humanos, procesos y datos.

La finalidad de los sistemas de información es ayudar a los gerentes a tomar decisiones acertadas, optimizar los procesos comerciales, aumentar la productividad y

[1] Las pruebas del sistema se deben realizar antes de hacer la implementación del mismo, para verificar que se cumpla con todas las necesidades del cliente.

brindar información relevante a los usuarios. Cada sistema de TI está compuesto por diversos recursos interrelacionados e interactivos, que se organizan de forma conveniente según su propósito específico. Algunos ejemplos pueden ser la recopilación de datos personales, el procesamiento del sistema, la lista y clasificación de archivos, entre otros.

Los recursos de los sistemas de información pueden ser:

- *Recursos humanos:* Personal que reúnen diferentes destrezas.
- *Datos:* información de la empresa que esté relacionada con el sistema a desarrollarse.
- *Actividades:* Procedimientos, pasos a seguir, estaciones de trabajo, etc.
- *Recursos informáticos:* aquellos elementos de hardware y software que serán utilizados en el desarrollo e implementación del sistema.

1.2.1 SISTEMAS DE INFORMACIÓN EN LA EMPRESA MODERNA

Los sistemas de información en las empresas modernas son una herramienta clave para la gestión eficiente y eficaz de los procesos comerciales, la toma de decisiones informada y el éxito general de la empresa.

Ejemplo de sistemas de información presente en las empresas modernas:

- *Sistemas de gestión empresarial (ERP):* sistemas integrados que permiten a una empresa gestionar de forma eficaz los procesos empresariales, incluidas áreas como compras, ventas, inventario, finanzas, recursos humanos y gestión de la producción.
- *Sistemas de gestión de la cadena de suministro (SCM):* son sistemas que ayudan a las empresas a gestionar y optimizar sus procesos de cadena de suministro, incluyendo la planificación, ejecución, seguimiento y análisis de la cadena de suministro.
- *Sistemas de gestión del conocimiento (KM):* son sistemas que permiten a las empresas recopilar, almacenar, compartir y utilizar el conocimiento y la experiencia de sus empleados para mejorar la toma de decisiones y el rendimiento general del negocio.
- *Sistemas de gestión de relaciones con los clientes (CRM):* son sistemas que permiten a las empresas gestionar y analizar las interacciones con los

clientes, incluyendo ventas, marketing y servicios postventa, para mejorar la satisfacción y retención de los clientes.

- *Sistemas de análisis de datos y Business Intelligence (BI):* son sistemas que permiten a las empresas recopilar, analizar y visualizar datos para tomar decisiones informadas y mejorar el rendimiento general del negocio.

1.2.2 CARACTERÍSTICAS DE LOS SISTEMAS

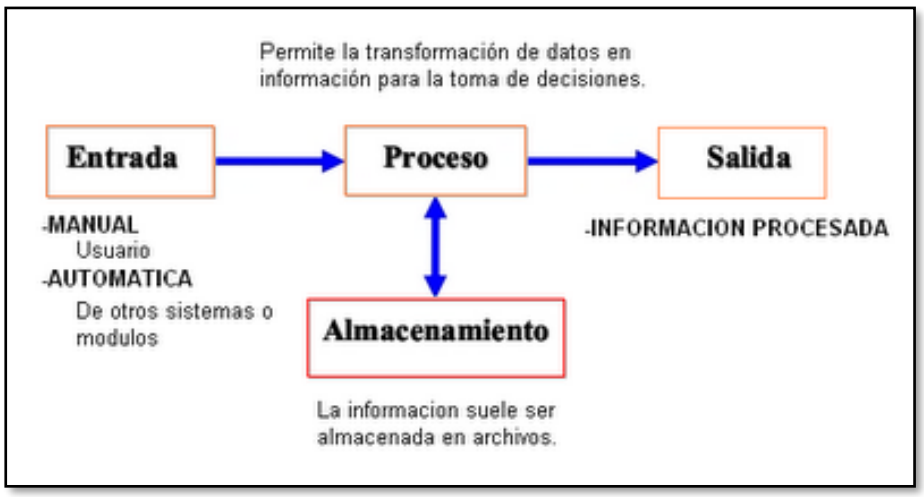
- *Partes que compone el sistema*: elementos o subsistemas existentes.
- *Objetivos:* razón de ser del sistema.
- *Medio ambiente:* elementos de hardware y software que interactúan con el sistema operativo y el sistema que se va a implementar.
- *Elemento de control:* son todos los estándares necesarios para desarrollar o actualizar un sistema de calidad.

Los ajustes del sistema se realizan al comparar el desempeño del sistema actual con estos estándares.

1.2.3 ACTIVIDADES DE UN SISTEMA DE INFORMACIÓN:

Figura 1

Actividades de un sistema de información



Nota. La figura representa las actividades que realiza un sistema de información, desde la entrada de los datos, su proceso que se realiza a esos datos, así como el almacenamiento para posteriormente tener una salida de información.

Fuente: https://sistinfouan.weebly.com/actividades-baacutesicas-de-un-sistema-de-informacioacuten.html

Entrada: proceso mediante el cual se captura y prepara la información para su posterior procesamiento. Generalmente constituidas por:

- *Datos:* descripciones básicas de cosas, acontecimientos, actividades y transacciones que se registran, clasifican y almacenan pero que no se organizan de acuerdo con ningún significado específico.
- *Programas:* aplicaciones y recursos que permiten desarrollar diferentes tareas en un computador u otro equipo tecnológico.
- *Procedimientos:* serie de pasos bien definidos que permitan y faciliten la realización de un trabajo de la manera más adecuada.
- *Estándares:* son las reglas las cuales establecen los pasos a ser realizados para el uso de determinado sistema.
- *Controles de acceso:* aprobación de acceso, el sistema adopta la decisión de conceder o rechazar solicitudes de acceso a él.
- *Hardware:* conjunto de componentes que conforman la parte física de un computador.

Almacenamiento: almacenamiento de datos e información de forma organizada para su posterior uso. Para hacer fácil dicha recuperación, los datos se organizan en:

- *Campo:* agrupación de caracteres que identifican a un sujeto, lugar u objeto, por ejemplo: nombre de un cliente.
- *Registro:* conjunto de campos interrelacionados, por ejemplo, el registro de nómina el cual se componen de nombre, ítem, departamento y sueldo.
- *Archivo:* conjunto de registros interrelacionados, por ejemplo, archivos pagos del mes de enero del 2022, similar a una base de datos.
- *Base de datos:* conjunto de registros interrelacionados.

Proceso: capacidad de efectuar operaciones con los datos guardados en las unidades de memoria. Capacidad del Sistema de Información (SI) para convertir la información

procesada o datos de entrada en información para el exterior[2]. Durante el procesamiento se evidencia lo siguiente:

- Aumenta, manipula y organiza la forma de los datos.
- Analiza y evalúa su contenido.
- Selecciona la información para ser usada en la toma de decisiones, constituyendo un componente clave en el SI gerencial.

El resultado de la información puede ser utilizada para la toma de decisiones empresariales que pueden afectar en algún cambio.

Resultados: son los resultados e información generados por el sistema. Para muchos usuarios finales la salida, es la única razón para el desarrollo del sistema y la base sobre la que ellos evalúan la utilidad de la aplicación. Para diseñar una salida se debe:

- Determinar qué información presentar.
- Decidir si la información será presentada en forma visual o impresa y seleccionar el medio de salida.
- Disponer la presentación de la información en un formato aceptable.
- Decidir cómo distribuir la salida entre posibles destinatarios.

Ejemplo de las actividades de un sistema de información:

Descubrir las actividades de un sistema de información para la gestión de datos de los clientes de la empresa

Entradas:

- Datos generales del cliente: nombre, dirección, tipo cliente, etc.
- Políticas de crédito: límite de crédito, plazo fijo
- Facturas
- Pagos, depuraciones, etc.

Proceso:

- Cálculo de antigüedad de saldos

[2] Cuando se habla del exterior se puede hacer énfasis en el resultado de la información ya procesada. Por ejemplo, un estado de matrícula, una factura, etc.

- Cálculo de interés
- Cálculo de saldo de un cliente

Almacenamiento:

- Movimientos del mes
- Catálogo del cliente
- Facturas

Salida:

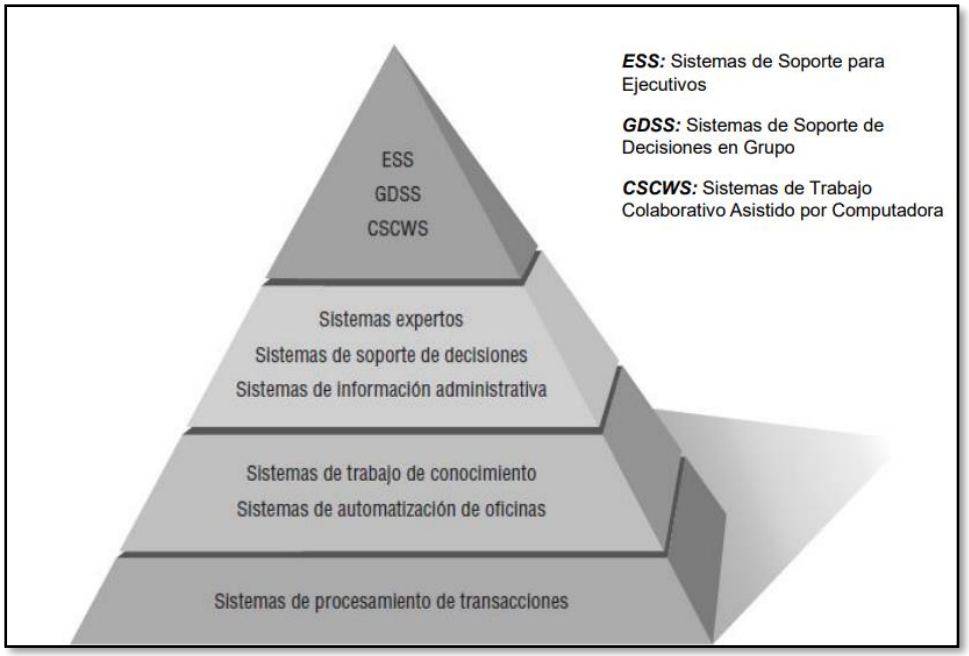
- Reporte de pagos
- Estados de cuentas
- Pólizas contables
- Consultas de saldo

1.2.4 TIPOS DE SISTEMAS DE INFORMACIÓN

Diferentes organizaciones tienen diferentes sistemas de información para las mismas áreas funcionales. Como no hay dos organizaciones con exactamente los mismos objetivos, estructuras o intereses, los sistemas de información deben ser hechos a medida para atacar las características únicas de cada una. (Sedici, 2023)

Figura 2

Tipos de sistemas de información



Nota. En la figura podemos observar los diferentes tipos de sistemas de información, siendo la base el sistema de procesamiento de transacciones.
Fuente: https://simplesoftmx.blogspot.com/2013/10/tipos-de-sistemas.html

Figura 3

Niveles de los tipos de sistemas de información

Nota. En la figura podemos observar los diferentes niveles de los tipos de información al igual que a los grupos que pertenece según sus áreas funcionales.
Fuente de la imagen:
http://nestor-omana-sistemasinformacion.blogspot.com/2010/05/tipos-de-sistemas-segun-el-nivel-de.html

La organización está dividida en los siguientes niveles:

- Estratégico.
- Administrativo.
- Del conocimiento.
- Operativo.

Estos a su vez, se dividen en áreas funcionales tales como marketing, manufactura, finanzas, contabilidad y recursos humanos. Los sistemas se construyen para servir esos diferentes intereses de la organización. Se pueden reconocer cuatro tipos principales de sistemas de información que sirven a los diferentes niveles de la organización: sistemas a nivel operativo, sistemas a nivel de conocimiento, sistemas a nivel administrativo y sistemas a nivel estratégico. Los sistemas a nivel operativo son sistemas que monitorean las actividades y transacciones elementales de la organización, tal como ventas, ingresos, depósitos, decisiones para asignar créditos, y el flujo de materiales en una empresa. (Sedici, 2023)

Desde un punto de vista empresarial u organizativo, los sistemas de información pueden clasificarse en:

- *Sistema de apoyo a ejecutivos (ESS):* ayuda a los ejecutivos a organizar sus interacciones con el mundo exterior al proporcionar gráficos y tecnología de comunicaciones en lugares de fácil acceso, como salas de juntas o salas de conferencias, oficinas privadas de la empresa. Además, es un sistema de nivel estratégico que cubre áreas como la previsión de tendencias de ventas, la planificación operativa, la previsión presupuestaria, las comunicaciones y la planificación de recursos humanos.

Tabla 1

Especificaciones del sistema de apoyo a ejecutivos

Sistemas a Nivel Estratégico				
Pronóstico de tendencias de ventas a largo plazo	Plan de operaciones a largo plazo	Pronóstico de presupuesto a largo plazo	Planificación de ganancias	Planificación de mano de obra

Nota. En la tabla podemos observar las especificaciones del sistema en el nivel estratégico
Fuente:
http://sedici.unlp.edu.ar/bitstream/handle/10915/4060/I_Sistemas_de_informaci%C3%B3n.pdf?sequence=5&isAllowed=y

- *Sistemas de Procesamiento de Transacciones (TPS)*: También conocidos como sistemas de gestión operativa, recopilan la información pertinente a las transacciones de la empresa, es decir, de su funcionamiento.

Tabla 2

Sistema de procesamiento de transacciones

Sistemas a Nivel Operativo				
	Control de máquinas	Seguridades del Comercio	Payroll (nómina)	Indemnizaciones
Seguimiento de órdenes	Scheduling de producción		Cuentas a pagar	Entrenamiento y desarrollo
Procesamiento de órdenes	Control de movimiento de material	Administración de caja	Cuentas por cobrar	Mantención de los registros de empleados
Ventas y Marketing	Manufactura	Finanzas	Contabilidad	Recursos Humanos

Nota. En la tabla se puede observar las especificaciones del sistema de procesamiento en su nivel operativo.
Fuente:
http://sedici.unlp.edu.ar/bitstream/handle/10915/4060/I_Sistemas_de_informaci%C3%B3n.pdf?sequence=5&isAllowed=y

Tabla 3

Aplicaciones del sistema de procesamiento de transacciones

Tipos de TPS	Principales funciones del sistema	Principales sistemas de aplicación
Sistemas de Venta/Marketing	Administración de ventas Búsqueda de mercados Promoción Precios Nuevos productos	Sistemas de información de órdenes de ventas Sistemas de búsqueda de mercados Sistemas de precios
Sistemas de Manufactura/Producción	*Scheduling* *Compras* *Embarque/recepción* *Ingeniería* *Operaciones*	*Sistemas de planificación de recursos* *Sistema de planificación de operaciones* *Sistemas de control de órdenes de compra* *Sistemas ingenieriles* *Sistemas de control de calidad*
Sistemas de Finanzas/Contabilidad	Presupuesto Contabilidad Facturación Costos	Sistemas de contabilidad Cuentas por cobrar y por pagar Presupuestos Sistemas de administración de fondos
Sistemas de Recursos Humanos	Registros de personal Beneficios Indemnizaciones Relaciones laborales Entrenamiento	Payroll Registros de empleados Sistemas de beneficios Sistemas de ascensos del personal
Otros tipos (ej. universidad)	Admisión Registro de grados Registro de cursos Alumnado	Sistemas de registración Sistema de inscripción de alumnos Sistemas de control de clasificación de currículum Sistema de beneficios de alumnos

Nota. En la tabla se puede observar la identificación algunas aplicaciones típicas de un TPS. La tabla muestra que hay cinco categorías funcionales de TPS: ventas/marketing, manufactura/producción, finanzas/contabilidad, recursos humanos, y otros tipos de TPS que son únicos a una industria particular.
Fuente:
http://sedici.unlp.edu.ar/bitstream/handle/10915/4060/I_Sistemas_de_informaci%C3%B3n.pdf?sequence=5&isAllowed=y

- *Sistemas de Información Ejecutiva (EIS):* Realiza un seguimiento de las variables de gestión en un área particular de la empresa a partir de su información interna y externa.
- *Sistemas de Información Gerencial (MIS):* Contemplan la información general de la empresa y la comprenden como un todo. Además, se encuentra a nivel administrativo y apoya en procesos en tales como: administración de ventas, control de inventarios, elaboración de presupuestos anuales, análisis de inversión capital y análisis de reubicación. (Omaña, 2010).

Las características de estos sistemas son las siguientes:

- Soportan decisiones estructuradas a los niveles de control administrativo y operativo. Sin embargo, son útiles para los gerentes senior para propósitos de planificación.
- Son orientados al control y reporte. Son diseñados para generar reportes sobre operaciones existentes y para realizar el control diario de las operaciones.
- Utilizan datos corporativos existentes y flujos de datos.
- Tienen poca capacidad analítica.
- Ayudan generalmente a la toma de decisiones usando datos pasados y presentes.
- Son relativamente poco flexibles.
- Tienen una orientación interna más que externa.

- *Sistemas de soporte de decisiones (DSS):* se centra en el procesamiento de la información dentro y fuera de la organización para apoyar la gestión empresarial. Además, se ubica a nivel administrativo y permite realizar análisis de ventas por región, programación de programas de referidos, análisis de costos, análisis de precios y utilidades, así como análisis de valor de contratos.

Tabla 4

Especificaciones del sistema de soporte de decisiones

Sistemas a Nivel Gerencial				
Administra-ción de ventas	Control de inventario	Presupuesto anual	Análisis de inversión de capital	Análisis de reubicación
Análisis de las regiones de ventas	Planifica-ción de la producción	Análisis de costos	Análisis precio/renta bilidad	Análisis de costos contractuales

Nota. En la tabla podemos observar las especificaciones del sistema de soporte de decisiones que se debe tener en cuenta para el desarrollo de sistemas de información.

Las características de los DSS son las siguientes:

- Ofrece a los usuarios flexibilidad, adaptabilidad y rápida respuesta.
- Opera con poca asistencia desde los profesionales de la computación.

- Provee soporte para decisiones y problemas cuyas soluciones no pueden ser especificadas en forma prematura.
- Usa análisis de datos y herramientas de modelado sofisticados

- *Sistema de trabajo de conocimiento (KWS):* apoyan a profesionales como científicos, ingenieros y médicos ayudándolos a generar conocimiento e integrar ese conocimiento en su empresa o sociedad. Además, es un sistema de nivel de conocimientos y entre las posibilidades que incluye puedes crear trabajos de ingeniería, gráficos y gestión.

Tabla 5

Sistema de trabajo de conocimiento

Sistemas a Nivel de Conocimiento		
Puestos de trabajo ingenieriles	Puestos de trabajo gráficos	Puestos de trabajo gerenciales
Procesamiento de palabras	Producción de imágenes documentales (document imaging)	Calendarios electrónicos

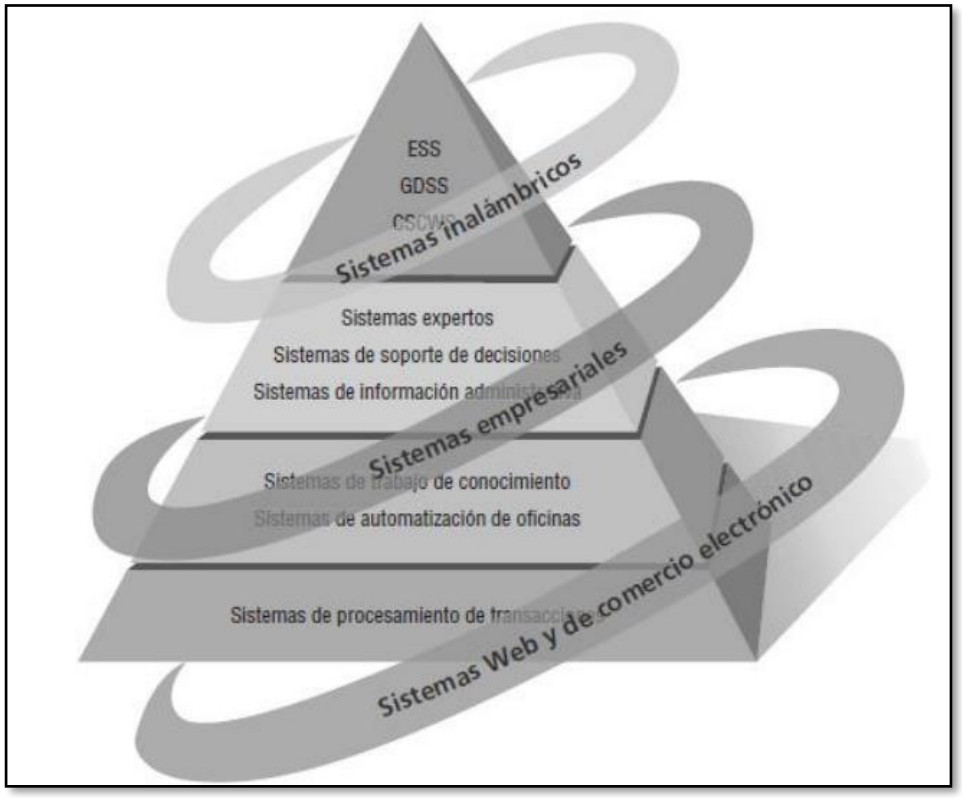
Nota. En la tabla se puede observar las especificaciones del sistema de trabajo de conocimiento. Fuente: Autores.

Existen otras formas especializadas o aplicadas de SI, dependiendo del campo puntual y de las funciones específicas que se esperan de cada uno. Sería demasiado extenso enlistarlas a todas.

Integración de las tecnologías en los sistemas:

Figura 4

Sistemas de información con integración en las tecnologías



Nota. En la figura podemos observar los tipos de sistemas de información con la integración de las diferentes tecnologías. Fuente: *http://ramcesdj1.blogspot.com/*

Sistemas de Web y de comercio electrónico: Hay muchos beneficios relacionados con el proceso de montar o mejorar una aplicación en Web:

- Aumenta el número de usuarios que se enteran de la disponibilidad de un servicio, producto, industria, persona o grupo.
- Los usuarios tienen la posibilidad de acceder las 24 horas del día.
- Se puede mejorar la utilidad y capacidad de uso del diseño de la interfaz.
- Se puede expandir un sistema globalmente en vez de permanecer en el entorno local, con lo cual se puede establecer contacto con personas en ubicaciones remotas sin preocuparse por la zona horaria en la que se encuentren. (Baltazar, 2008)

Sistemas empresariales: Algunos autores describen la integración como arquitectura orientada a servicios (SOA), la cual existe en capas. Los sistemas empresariales conformarían la capa superior. Estos sistemas, también conocidos como sistemas de planificación de recursos empresariales (ERP), están diseñados para llevar a cabo esta integración.

Sistemas para dispositivos inalámbricos y móviles: Por ejemplo, mediante el uso de la tecnología pull, un agente inteligente puede buscar en la Web historias de interés para el usuario después de haber observado sus patrones de comportamiento a través del tiempo, y realizar búsquedas en la Web sin tener que estar solicitándole información en forma continua.

Software de código fuente abierto: esta es una alternativa al desarrollo de software tradicional donde el código propietario está oculto para los usuarios. Con el software de código abierto, los usuarios y desarrolladores pueden estudiar, compartir y modificar el código o las instrucciones de la computadora. Estas reglas de la comunidad asumen que cualquier cambio en los programas debe estar disponible para todos los participantes del proyecto.

El uso extendido de OSS puede ayudar a aliviar la severa escasez de programadores, al poner las herramientas de programación en manos de estudiantes de países en desarrollo en menos tiempo del que se requeriría si estuvieran limitados al uso de paquetes propietarios, y puede ayudar a resolver grandes problemas mediante una colaboración intensa y extensa. (Baltazar, 2008)

Es el soporte, servicio y comercialización de productos relacionados con la integración de Sistemas. Estos productos y servicios, ofrecen las nuevas tecnologías que le ayuden a obtener soluciones reales a la empresa. Algunos de ellos son:

- Sistema de planeación de recursos empresariales, sistemas de planificación de recursos empresariales.
- Sistema para dispositivos inalámbricos y portátiles.
- Sistemas basados en la web.
- Comercio electrónico.
- Sistema de código abierto

1.3 ROLES

Los roles del sistema son grupos de usuarios que tienen asignados un conjunto específico de permisos para llevar a cabo tareas particulares. Estos roles se otorgan a usuarios específicos con el objetivo de asegurar que cada uno tenga los permisos adecuados para realizar sus labores.

Algunos ejemplos comunes de roles del sistema incluyen Administrador, Usuario, Invitado, Desarrollador, Analista y Experto, aunque pueden variar dependiendo del sistema en cuestión. Aunque los nombres y propósitos de los roles pueden ser similares en diferentes sistemas, en ocasiones pueden diferir en función de las necesidades específicas de cada uno.

1.3.1 ROL DEL ANÁLISIS DE SISTEMAS

El rol del analista del sistema consiste en evaluar cómo los usuarios interactúan con la inteligencia artificial (IA), a través del examen detallado de los procesos de entrada y salida de datos. Además, el analista asigna los roles correspondientes a cada usuario, en función de la funcionalidad que se requiere para permitirles acceder tanto a la IA como a la información relacionada.

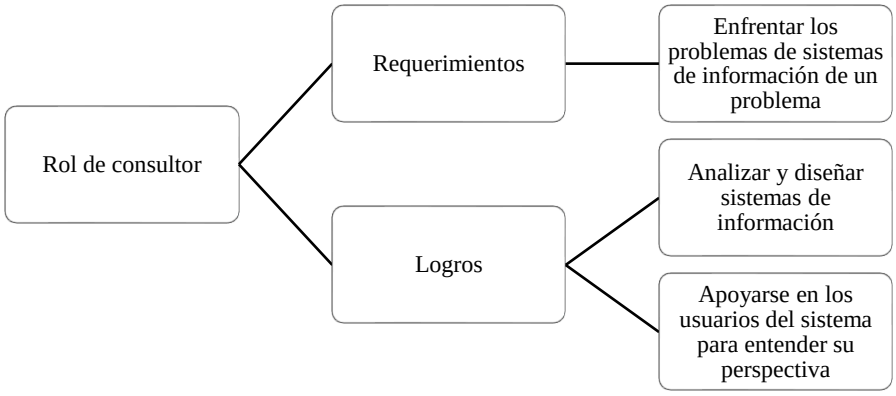
De esta manera, el analista del sistema es clave en la gestión efectiva de la IA dentro de una organización, al asegurarse de que los usuarios tengan los permisos adecuados para acceder a la información relevante y utilizar la IA de manera eficiente.

Rol de consultor

Es fundamental que el analista de sistemas conozca la naturaleza del negocio y su relación con la tecnología a través de los sistemas de información (SI). Para lograr esto, el analista debe trabajar de cerca con los usuarios de los SI, quienes pueden proporcionar información valiosa para entender la cultura organizacional desde el punto de vista de la empresa.

Figura 5

El analista en su rol de consultor



Nota. En la figura podemos observar los requerimientos y logros que deben cumplir el analista de sistema en su rol de consultor. Fuente: Autores

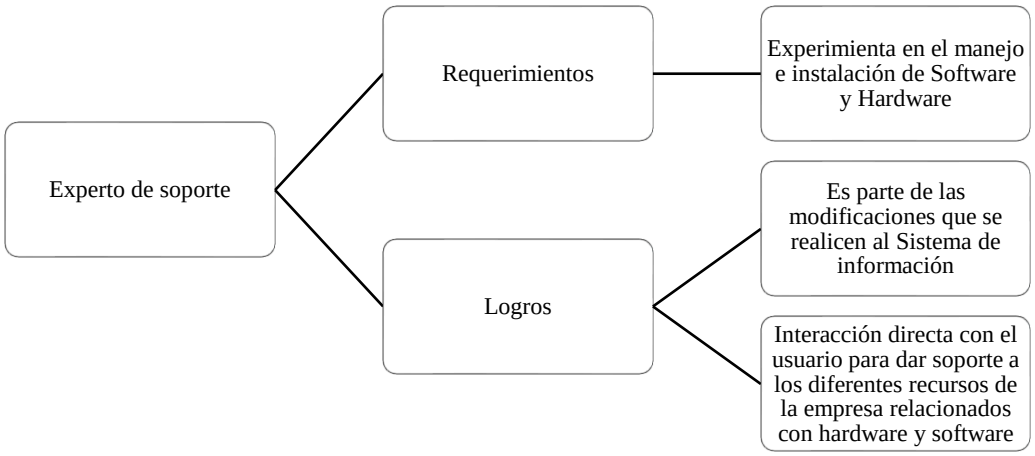
Roles de experto de soporte

El rol del profesional en hardware y software se basa en su amplia experiencia en estas áreas. Aunque no se dedica a la administración directa de proyectos, es un recurso clave para aquellos que sí lo hacen.

Este profesional suele estar involucrado en la realización de pequeñas modificaciones que afectan a un departamento específico. Al colaborar con otros miembros del equipo, puede brindar soluciones a los desafíos tecnológicos que surgen en el día a día de la empresa.

Figura 6

El analista en su rol de experto de soporte



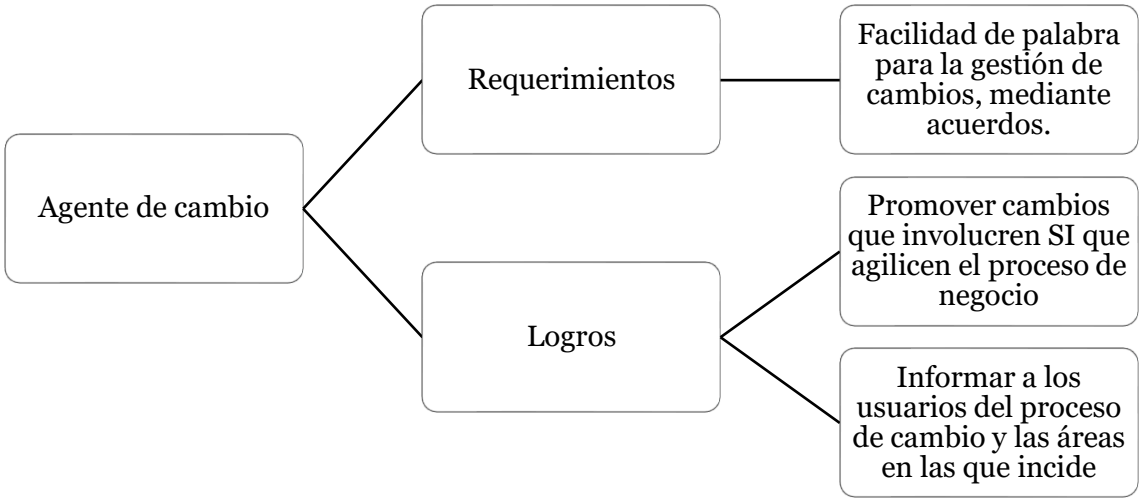
Nota. En la figura podemos observar los requerimientos y logros que debe cumplir el analista en su rol de experto de soporte. Fuente: Autores.

Rol de agente de cambio

Desarrolla un plan de cambio y trabaja con otros para facilitarlo. Rol con mayor responsabilidad, ya que implica cambio en procesos. Instruye a los usuarios sobre cambios realizados.

Figura 7

El analista de sistemas en su rol de agente de cambio



Nota. En la figura podemos observar los requerimientos y logros del analista de sistema en su rol de agente de cambio. Fuente: Autores

Características del analista de sistemas

- El analista debe ser capaz de trabajar con personas de todo tipo y tener experiencia en cuanto al trabajo con computadoras.
- El analista de sistemas tiene que lidiar con muchos roles a la vez, y también puede asumir varios roles.
- Los tres principales roles del analista son como: consultor, experto de soporte y agente de cambios.
- Un analista es un solucionador de problemas

1.3.2 ROL DEL DESARROLLADOR DE SOFTWARE

Los Desarrolladores de Software o de Sistemas trabajan para firmas y empresas dedicadas al ramo de la Computación. Se encargan de desarrollar los pilares de los sistemas operativos que son creados por los Programadores y de probar el código de nuevos programas para garantizar su eficiencia. Asimismo, realizan pruebas de calidad en nuevos proyectos antes de su lanzamiento. (Neuvoo, 2017)

A continuación, las funciones más comunes de un desarrollador de Software:

- *Analista:* Responsable de recopilar requisitos de software, de analizar y documentar los requisitos y de diseñar la arquitectura del sistema.
- *Programador:* Responsable de codificar y depurar el código para crear la aplicación.
- *Administrador de sistemas:* Responsable de instalar, configurar y mantener el entorno de ejecución de la aplicación.
- *Tester:* Responsable de realizar pruebas de software para garantizar su funcionalidad y estabilidad.
- *Diseñador de interfaz de usuario:* Responsable de crear una interfaz de usuario intuitiva y atractiva para el usuario final.
- *Documentador:* Responsable de documentar el software para usuarios finales y otros desarrolladores.
- *Investigador:* Responsable de investigar nuevas tecnologías y herramientas para mejorar el proceso de desarrollo de software.

Labores diarias del Desarrollador de Software

- Reunirse con los clientes y Gerentes de Proyecto para diseñar y desarrollar nuevos programas.
- Establecer parámetros y diseñar la arquitectura de nuevos programas.
- Diseñar, escribir, leer, probar y corregir el código de nuevos programas.
- Realizar pruebas de medición de calidad y detectar errores en el desarrollo del programa.
- Preparar la documentación necesaria para programas nuevos o actualizados.

1.4 REQUERIMIENTOS DEL SOFTWARE

El software debe ejecutar la aplicación de manera eficiente en los principales sistemas operativos, como Windows, Mac OS X y Linux, y ser compatible con la mayoría de los navegadores web modernos. Además, debe ser fácil de instalar y configurar, e incluir herramientas de seguridad para proteger la información de los usuarios. La interfaz debe ser fácil de usar y la configuración debe ser intuitiva. El software debe ser escalable para

satisfacer las necesidades crecientes de los usuarios y debe tener una función de administración para administrar y monitorear el software. También debe contener una funcionalidad de diagnóstico y soporte para ayudar al usuario a detectar y resolver problemas, así como una función de actualización. Durante el desarrollo de requisitos, es importante tener una separación clara entre los diferentes niveles de descripción, utilizando el término "requisitos del usuario" para requisitos abstractos de alto nivel y "requisitos del sistema" para describir en detalle lo que debe hacer el sistema.

El termino requerimiento no se utiliza de forma consistente en la industria del software, Un requerimiento se visualiza como una declaración abstracta de alto nivel de un servicio que debe proveer el sistema o como una restricción de éste. (Sommerville, 2005)

1.4.1 TIPOS DE REQUERIMIENTOS

Requerimiento del Usuario: son declaraciones en lenguaje natural y diagramas, de los servicios que se espera que el sistema provea y de las restricciones bajo las cuales debe operar.

Los requerimientos del sistema: establecen con detalle los servicios y restricciones del sistema. El documento de requerimientos del sistema, algunas veces denominado especificaciones funcionales, debe ser preciso. Este sirve como un contrato entre el comprador del sistema y el desarrollador de software.

Especificaciones del diseño del software: es una descripción abstracta del diseño del software que es una base para un diseño e implementación detallados. (Sommerville, 2005)

Etapas de la fase de requerimientos

- *Obtención de requerimientos:* búsqueda y obtención de los requerimientos desde los grupos de interés.
- *Análisis:* comprobación de la consistencia de los requerimientos.
- *Verificación:* constatación de que los requerimientos especificados son correctos.

Características que deben cumplir los requerimientos:

- *Actual:* el requerimiento no debe volverse obsoleto con el paso del tiempo.

- *Cohesión:* el requerimiento debe dirigirse a solo una única cosa.
- *Completo:* el requerimiento debe estar completamente declarado en un único lugar, sin información faltante.
- *Consistente:* el requerimiento no debe contradecir ningún otro requerimiento y debe ser completamente consistente con toda la documentación.
- *Correcto/necesario:* el requerimiento debe cumplir con la necesidad declarada por los interesados en el sistema/software.
- *Factible/viable:* el requerimiento debe poder ser implementado.
- *No ambiguo:* el requerimiento debe estar conscientemente declarado. Debe expresar hechos objetivos, no opiniones subjetivas. Debe poder ser interpretado de una única manera.
- *Obligatorio:* el requerimiento debe representar una característica definida por el grupo interesado en el desarrollo del sistema/software, su ausencia no puede ser reemplazada.
- *Observable externamente:* el requerimiento debe especificar una característica observable externa o experimenta por el usuario del producto.
- *Verificable/demostrable:* la implantación del requerimiento debe poder ser resuelta en alguno de estos cuatro métodos: inspección, análisis, demostración o prueba. (Puentes, 2015)

1.4.2 REQUERIMIENTOS FUNCIONALES

Los requisitos funcionales son declaraciones sobre los servicios que proporcionará el sistema sobre cómo responderá a ciertas entradas. Cuando hablamos de entrada, no necesariamente nos referimos solo a la entrada del usuario. Esto podría ser interacción con otros sistemas, respuestas automatizadas, procesos predefinidos. En algunos casos, los requisitos funcionales del sistema también definen claramente lo que el sistema no debe hacer. Es importante recordar: RF también puede ser negativo. Mientras su comportamiento resulte en una respuesta funcional a otro usuario o sistema, seguirá siendo cierto. Además, no solo es cierto, sino que también es necesario definirlo. Y eso nos lleva al siguiente punto.

Los requisitos funcionales son declaraciones de los servicios que prestará el sistema, en la forma en que reaccionará a determinados insumos. Cuando hablamos de las entradas, no necesariamente hablamos sólo de las entradas de los usuarios. Pueden ser

interacciones con otros sistemas, respuestas automáticas, procesos predefinidos. En algunos casos, los requisitos funcionales de los sistemas también establecen explícitamente lo que el sistema no debe hacer. Es importante recordar esto: un RF[3] puede ser también una declaración negativa. Siempre y cuando el resultado de su comportamiento sea una respuesta funcional al usuario o a otro sistema, es correcto. Y más aún, no sólo es correcto, sino que es necesario definirlo. Y eso nos lleva al siguiente punto. (Requeridos Blog, 2018)

Los requisitos funcionales de un software se suelen registran en la matriz de trazabilidad de requerimientos y en la especificación de requerimientos de software, este último, documenta las operaciones y actividades que el sistema debe poder desempeñar.

Entre los posibles requerimientos funcionales de un sistema, se incluyen:

- Descripciones de los datos a ser ingresados en el sistema.
- Descripciones de las operaciones a ser realizadas por cada pantalla.
- Descripción de los flujos de trabajo realizados por el sistema.
- Descripción de los reportes del sistema y otras salidas.
- Definición de quien puede ingresar datos en el sistema.
- Como el sistema cumplirá los reglamentos y regulaciones de sector o generales que le sean aplicables.

Al igual que otros tipos de requerimientos de software, como por ejemplo los requerimientos no funcionales, los requerimientos funcionales se pueden clasificar según su finalidad, como por ejemplo requerimientos de negocio, requerimientos originados en aspectos regulatorios, de seguridad, entre otros. (Eriksson, 2017)

1.4.3 REQUERIMIENTOS NO FUNCIONALES

Son aquellos requerimientos que no se refieren directamente a las funciones específicas que entrega el sistema, sino a las propiedades emergentes de éste como la fiabilidad, la respuesta en el tiempo y la capacidad de almacenamiento. De forma alternativa, definen las restricciones del sistema como la capacidad de los dispositivos de entrada/salida y la representación de datos que se utiliza en la interface del sistema. (Vera, 2021)

[3] Los requerimientos funcionales deben estar escritos en leguaje natural, ya sea positivamente o negativamente

Los requisitos no funcionales surgen de diversas fuentes, como las necesidades del usuario, las restricciones presupuestarias, las políticas de la empresa, la interacción con otros sistemas de software o hardware, o factores externos como las políticas de privacidad, seguridad y regulaciones.

Ventajas de los requerimientos no funcionales

- Ayudan a garantizar que el sistema satisfaga las necesidades del usuario
- Garantiza que el sistema sea adecuado para su propósito.
- Garantiza que el sistema sea escalable, seguro y confiable
- Garantiza que el sistema sea fácil de usar y mantener. (Visure Solutions, 2023)

Ejemplos de requerimientos no funcionales

- *Seguridad:* El sistema debe garantizar la seguridad de los datos y procesos de la empresa.
- *Rendimiento:* El sistema debe tener un tiempo de respuesta aceptable para usuarios remotos y locales.
- *Uso:* El sistema debe ser sencillo de usar y entender para todos los usuarios.
- *Disponibilidad:* El sistema debe estar disponible las 24 horas del día, los 7 días de la semana.
- *Escalabilidad:* El sistema debe ser capaz de escalar para satisfacer las necesidades del usuario.
- *Mantenimiento:* El sistema debe ser fácilmente mantenerle a lo largo de su vida útil.
- *Estabilidad:* El sistema debe establecer y mantener una plataforma estable para todos los usuarios.
- *Integración:* El sistema debe ser compatible con los sistemas existentes de la empresa.

RESUMEN DEL CAPÍTULO 1

El análisis de sistemas de información es un proceso de investigación que se utiliza para identificar y entender los problemas que existen dentro de un sistema de información. Esto ayuda a los diseñadores de sistemas a mejorar y optimizar el sistema de información para que pueda cumplir con los objetivos de la empresa. El análisis de sistemas de información incluye la identificación de los problemas del sistema, la comprensión de los procesos actuales del sistema, el diseño de nuevos procesos para mejorar el rendimiento del sistema y la implementación de dichos procesos. También se identifican las herramientas y los recursos necesarios para garantizar el funcionamiento óptimo del sistema. El análisis de sistemas de información es una de las principales herramientas para asegurar que un sistema funcione de manera eficiente y eficaz.

El análisis de sistemas de información es un proceso de descubrimiento de los requisitos de un sistema de información. Implica el análisis de los requerimientos funcionales y no funcionales, así como la identificación de los procesos y procedimientos necesarios para la implementación de un sistema de información eficaz. El objetivo es mejorar la calidad y la eficiencia de los sistemas de información, para apoyar a la toma de decisiones y mejorar los resultados de la empresa. Esto se logra mediante el análisis de los problemas actuales y el diseño de una solución. El análisis de sistemas de información incluye la identificación de los requerimientos de los usuarios, la definición de los procesos de negocio, la selección de la tecnología adecuada, el diseño de la arquitectura del sistema y la evaluación de los riesgos. El análisis de sistemas de información se lleva a cabo para asegurar que los sistemas sean eficientes, flexibles, seguros y escalables.

CAPÍTULO 2
CICLO DE VIDA DE DEL DESARROLLO DE SISTEMAS

2.1 CICLO DE VIDA DE DEL DESARROLLO DE SISTEMAS

La metodología del Ciclo de Vida del Desarrollo de Software (SDLC) es un proceso sistemático que describe los pasos necesarios para llevar a cabo un proyecto de desarrollo de software, desde su inicio hasta su finalización. Esta metodología consta de diferentes fases, que generalmente incluyen la planificación, análisis, diseño, implementación, prueba y mantenimiento del software. El SDLC se utiliza como un marco de referencia para guiar a los desarrolladores de software durante el desarrollo de una aplicación o sistema, lo que permite al equipo entregar software de calidad en el plazo establecido.

Cualquier sistema de información va pasando por una serie de fases a lo largo de su vida. Su ciclo de vida comprende una serie de etapas entre las que se encuentran las siguientes:

- Planificación
- Análisis
- Diseño
- Implementación
- Pruebas
- Instalación o despliegue
- Uso y mantenimiento

Estas etapas son un reflejo del proceso que se sigue a la hora de resolver cualquier tipo de problema. Ya en 1945, mucho antes de que existiese la Ingeniería del Software, el matemático George Polya describió este proceso en su libro How to solve it (el primero que describe la utilización de técnicas heurísticas en la resolución de problemas). Básicamente, resolver un problema requiere:

- Comprender el problema (análisis)
- Plantear una posible solución, considerando soluciones alternativas (diseño)
- Llevar a cabo la solución planteada (implementación)
- Comprobar que el resultado obtenido es correcto (pruebas)

Las etapas adicionales de planificación, instalación y mantenimiento que aparecen en el ciclo de vida de un sistema de información son necesarias en el mundo real porque

el desarrollo de un sistema de información conlleva unos costes asociados (lo que se hace necesaria la planificación) y se supone que, una vez construido el sistema de información, éste debería poder utilizarse (si no, no tendría sentido haber invertido en su desarrollo). Para cada una de las fases en que hemos descompuesto el ciclo de vida de un sistema de información se han propuesto multitud de prácticas útiles, entendiendo por prácticas aquellos. (Berzal, 2023)

2.1.1 IDENTIFICACIÓN DEL PROBLEMA

En esta etapa, se identifican y documentan los desafíos actuales y se establecen los objetivos y requisitos del proyecto. Definir el problema puede implicar entrevistar a los usuarios finales, revisar la documentación existente, evaluar la tecnología actual y detectar cualquier brecha entre las necesidades del usuario y la funcionalidad actual del sistema.

Delimitación del ámbito del proyecto

Resulta esencial determinar el ámbito del proyecto al comienzo del mismo. Han de establecerse de antemano cuales son los problemas a resolverse durante la realización del proyecto y cuáles se dejarán fuera. Tan importante es determinar los aspectos abarcados por el proyecto como fijar aquellos aspectos que no se incluirán en el proyecto. Estos últimos han de indicarse explícitamente. Si es necesario, se puede especificar todo aquello que se posponga hasta una versión posterior del sistema. Si, en algún momento, fuese necesario incluir en el proyecto algún aspecto que no había sido considerado o que ya había sido descartado, es obligatorio reajustar la estimación del coste del proyecto y su planificación temporal. (Berzal, 2023)

Estudio de viabilidad

Con recursos ilimitados (tiempo y dinero), casi cualquier proyecto se podría llevar a buen puerto. Por desgracia, en la vida real los recursos son más bien escasos, por lo que no todos los proyectos son viables. En un conocido informe de 1994 (el informe Chaos del Standish Group), se hizo un estudio para determinar el alcance de la conocida como "crisis crónica de la programación" y, en la medida de lo posible, identificar los principales factores que hacen fracasar proyectos de desarrollo de software y los ingredientes clave que pueden ayudar a reducir el índice de fracasos. De entre los proyectos analizados:

- Sólo uno de cada seis se completó a tiempo, de acuerdo con su presupuesto y con todas las características inicialmente especificadas.
- La mitad de los proyectos llegó a completarse eventualmente, costando más de lo previsto, tardando más tiempo del estimado inicialmente y con menos características de las especificadas al comienzo del proyecto.
- Por último, más de un 30% de los proyectos se canceló antes de completarse

Antes de comenzar un proyecto, se debería evaluar la viabilidad económica, técnica y legal del mismo. Y no sólo eso, el resultado del estudio de viabilidad debería ajustarse a la realidad. A Jerry Weinberg, un conocido consultor, se le ocurrió preguntar a los asistentes a una conferencia suya, en el Congreso Internacional de Ingeniería del Software de 1987, cuántos de ellos habían participado en un estudio de viabilidad en el que se hubiese determinado que el proyecto no era técnicamente viable. De los mil quinientos asistentes, nadie levantó la mano. (Berzal, 2023)

Objetivo

Entrevistar a los encargados de la administración de usuarios, sintetizar el conocimiento obtenido, estimar el alcance del proyecto y documentar resultados.

Personas involucradas

Usuarios, analistas y administradores del sistema involucrados en el proyecto.

Resultado

Informe de viabilidad, el cual contiene la definición de un problema y sintetiza los objetivos.

2.1.2 DETERMINACIÓN DE LOS REQUERIMIENTOS

Objetivo

- Determinar las necesidades de los usuarios involucrados, mediante el uso de varias herramientas (entrevistas, muestreos, cuestionarios, prototipos)
- Comprender la forma en que interactúa el usuario con el SI actual.

- Diseñar un sistema fácil de usar, aprender y recordar.

Personas involucradas

Todas las personas que interactúan con el SI actual. Se establece fortalezas y limitaciones físicas, para que el sistema sea perceptible, legible y seguro.

Resultado

Comprensión de información que requieren los usuarios para realizar sus trabajos acordes a roles que desempeñan.

2.1.3 ANÁLISIS DE LAS NECESIDADES

Objetivo

- Uso de herramientas y técnicas especiales que ayudan a realizar la determinación de requerimientos como: diagramas de flujo de datos para graficar entradas, procesos y salida de las funciones de la empresa, diagramas de actividad o de secuencia de eventos.
- Desarrollo de diccionario de datos

Involucrados

Análisis de decisiones de estructuradas llevadas a cabo, para determinar condiciones, alternativas de condición, acciones y reglas de acción.

Resultados

Propuesta de sistemas en la que sintetiza todo lo que ha averiguado sobre los usuarios. Análisis de costo – beneficio.

2.1.4 DISEÑO DEL SISTEMA

Objetivo

Utiliza la información recolectada antes, para realizar el diseño lógico del sistema de información (procedimientos para que el usuario ingrese datos con precisión)

Involucrados

Desarrollo de interfaces para la conexión de usuarios con el sistema. Diseño de base de datos, Diseño de controles y procedimientos de respaldos para proteger el sistema y los datos.

Resultado

Trabajo con los usuarios para diseñar una salida (pantalla impresa) que cumpla con sus necesidades de información.

2.1.5 DESARROLLO Y DOCUMENTACIÓN

Objetivo

Los programadores desempeñan un rol clave en esta fase, ya que diseñan, codifican y eliminan los errores sintácticos de los programas de computadoras.

Involucrados

El analista trabaja con los programadores para desarrollar el software original requerido.

Resultados

Desarrollo de módulos funcionales[4] del SI. Documentación[5] que indica a los usuarios como deben usar el software y que debe hacer en caso de que ocurran problemas.

2.1.6 PRUEBA Y MANTENIMIENTO DEL SISTEMA

Objetivo

Antes de utilizar el sistema de información, se debe probar. Es mucho menos costoso detectar los problemas antes de entregar el sistema a los usuarios. Una parte del procedimiento que prueba es llevada a cabo por los programadores solos; la otra la realizan junto con los analistas de sistemas.

[4] El módulo del sistema de información no precisamente debe estar funcionando al 100%
[5] Esta documentación se puede englobar en el manual de usuario

2.1.7 IMPLEMENTACIÓN Y EVALUACIÓN DEL SISTEMA

El desarrollador junto con el analista de sistemas ayuda a la implementación del sistema de información. En esta fase hay que capacitar a los usuarios para operar el sistema. Además, el desarrollador se encarga de procesos como convertir los archivos de los formatos anteriores a los nuevos, o crear una base de datos, instalar equipo y llevar el nuevo sistema a producción.

2.2 SEGURIDAD EN EL CICLO DE VIDA DEL DESARROLLO DEL SOFTWARE

El ciclo de vida del desarrollo del software es un proceso crítico para el éxito de cualquier proyecto. Es importante tomar en cuenta la seguridad en cada etapa para garantizar que el software esté libre de vulnerabilidades y errores potenciales.

En la etapa de planificación, el equipo debe definir los requisitos de seguridad, establecer los estándares de seguridad adecuados y desarrollar un plan de seguridad para el proyecto. En la etapa de diseño, se debe tener en cuenta la seguridad en el diseño del software. Los desarrolladores deben estar conscientes de los riesgos de seguridad y responsables de la protección de los datos.

En la etapa de construcción, los desarrolladores deben usar herramientas de seguridad para comprobar los códigos del software, la autenticación y los controles de acceso. En la etapa de pruebas, se deben realizar pruebas exhaustivas para detectar cualquier vulnerabilidad de seguridad. Esto incluye pruebas de penetración, análisis de seguridad y pruebas de estrés.

Por último, en la etapa de implementación, los equipos de seguridad deben estar al tanto de los cambios en el software y realizar una auditoría para verificar la seguridad. Además, se deben implementar medidas de seguridad para prevenir ataques externos.

Grandes mitos y excusas flacas

- La seguridad de la aplicación es responsabilidad del programador
- Nadie sabe cómo funciona. Por ende, no la van a atacar.
- Si no se encontraron vulnerabilidades hasta ahora
- A nadie le interesa atacar nuestra aplicación
- La aplicación es segura porque corre detrás de un firewall
- La aplicación es segura porque usa encriptación
- Si no corre como administrador / root. No funciona

- No hay manera de explotarla.
- No hay tiempo para incluir seguridad.

Tendencia actual

Participación de seguridad informática desde el comienzo de todos los proyectos de desarrollo.

Incorporar seguridad a lo largo de todas las etapas del ciclo de vida del desarrollo del software.

- Análisis
- Diseño
- Codificación
- Testing
- Deployment[6]

Buenas prácticas para el diseño de una aplicación segura

- Reducción de superficie de ataque
- Criterio del menor privilegio
- Fallar de manera segura
- Criterio de defensa en profundidad
- Diseño seguro de mensajes de error
- Diseño seguro de autenticación
- Separación de privilegios
- Iteración "amigable" con firewalls
- Administración segura de la información sensible
- Diseño de auditoría y logging
- Análisis de riesgo.

[6] Se debe realizar la seguridad a nivel de implementación del sistema

Ejemplo

Un usuario accede en nombre del otro

- El usuario tenía contraseña débil
- El ataque capturo datos de autenticación de la red
    - La comunicación no estaba encriptada
    - El mecanismo de autenticación es vulnerable
- El sistema permitió un ataque de diccionario / fuerza bruta
- El sistema de validación / autenticación es defectuoso
    - El sistema de validación no falla de manera segura
    - El sistema de validación es vulnerable a ataques de "SQL injection"[7]
- El sistema permite armar peticiones con IDs[8] de otros usuarios. (Milano, 2007)

[7] La inyección de SQL en un tipo de ciberataque encubierto
[8] Códigos únicos de usuarios del sistema

RESUMEN DEL CAPÍTULO 2

El capítulo sobre el ciclo de vida del desarrollo de sistemas aborda el proceso completo de desarrollo de un sistema informático. El ciclo de vida del desarrollo de sistemas es un enfoque estructurado y disciplinado utilizado para guiar el proceso de creación de un sistema, desde la concepción hasta la implementación y mantenimiento.

El capítulo presenta varios modelos de ciclo de vida, incluyendo el modelo en cascada, el modelo en V, el modelo iterativo e incremental y el modelo en espiral. Cada modelo tiene sus propias ventajas y desventajas, y se puede elegir en función de los requisitos y objetivos específicos del proyecto.

El modelo en cascada es un enfoque secuencial en el que cada fase del proceso de desarrollo se lleva a cabo en orden, con una fase que debe completarse antes de pasar a la siguiente. El modelo en V es una extensión del modelo en cascada que se enfoca en la verificación y validación de cada fase. El modelo iterativo e incremental divide el proyecto en pequeñas iteraciones o ciclos y se enfoca en el desarrollo de funcionalidades específicas en cada ciclo. Finalmente, el modelo en espiral es un enfoque más flexible que combina la planificación, el diseño, la construcción y la evaluación en ciclos repetitivos.

El capítulo también cubre los principales procesos en cada fase del ciclo de vida, incluyendo la planificación, el análisis de requisitos, el diseño, la implementación, las pruebas y el mantenimiento. En cada fase, se identifican los objetivos, las actividades clave y los entregables.

En resumen, el capítulo sobre el ciclo de vida del desarrollo de sistemas proporciona una visión general del proceso completo de creación de un sistema informático, incluyendo los modelos de ciclo de vida y los procesos clave en cada fase. El conocimiento de estos procesos es esencial para desarrollar sistemas informáticos efectivos y eficientes.

CAPÍTULO 3
MODELOS DE PROCESOS Y METODOLOGÍAS

3.1 MODELOS DE PROCESOS

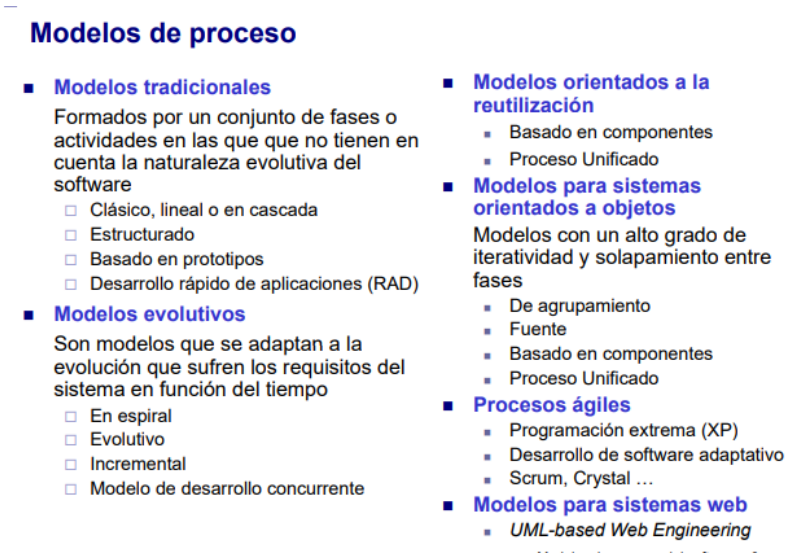
El análisis de un sistema aplicado a la informática está presente en cada organización o empresa que hace uso de los sistemas informáticos. Consideremos a un modelo (paradigma) de desarrollo de software como un enfoque sistemático y estructurado para planificar, diseñar, implementar y mantener software de alta calidad.

Sabemos que a las metodologías de desarrollo de software, se las puede definir como un conjunto de técnicas y métodos organizativos, que son utilizados para proponer soluciones de software informático. Estos modelos describen las actividades y tareas que deben realizarse en cada fase del ciclo de vida del desarrollo de software y establecen un conjunto de prácticas recomendadas para garantizar que el software cumpla con los requisitos del usuario, sea confiable y esté disponible para su uso. Entonces, Un modelo de desarrollo de software es una representación abstracta del Proceso de Desarrollo de software, y determina el orden en el que se llevan a cabo las actividades del proceso de desarrollo de software. (Gómez et al.,2019)

La historia indica que estos modelos han dado cierta estructura útil al trabajo de ingeniería de software y que constituyen un mapa razonablemente eficaz para los equipos de software. Existen varios modelos de desarrollo de software, cada uno con sus propias ventajas y desventajas.

Figura 8

Modelos de Procesos



Nota: en la figura "Modelos de Procesos" se detallan los modelos que se consideran para el desarrollo de un sistema. Fuente:http://avellano.usal.es/~mmoreno/ASTema2.pdf.

3.1.1 MODELOS TRADICIONALES

Los modelos tradicionales o conocidos también como modelos rígidos, en el desarrollo de sistemas, son aquellos que siguen una secuencia lineal y secuencial de fases. En este tipo de metodología, el desarrollador establece una disciplina de trabajo sobre el proceso de desarrollo, con el propósito de alcanzar un software más eficiente. Una de sus principales características es definir y establecer total y rígidamente todos y cada uno de los requisitos al inicio de los proyectos. Estas metodologías son poco flexibles y no permiten realizar cambios.

Dentro de lo cual, cabe mencionar que los modelos tradicionales hacen énfasis en la planificación total del proceso a implementar y una vez que está todo detallado, comienza el ciclo de desarrollo del software. Centrándose especialmente en el control del proceso, mediante una rigurosa definición de roles, actividades, herramientas, recursos y notaciones para el modelado y documentación detallada. Adicionalmente debemos tomar en cuenta que, las metodologías tradicionales no se adaptan adecuadamente a los cambios, por lo que no son métodos adecuados cuando trabajamos en un entorno, donde los requisitos no pueden predecirse o bien pueden presentar alguna variante en su desarrollo. Otra de las características importantes dentro de este enfoque, son los altos costes al implementar un cambio y la falta de flexibilidad en proyectos donde el entorno es volátil. (Maida, EG, Pacienzia, J. Metodologías de desarrollo de software [en línea]. Tesis de Licenciatura en Sistemas y Computación. Facultad de Química e Ingeniería "Fray

Rogelio Bacon". Universidad Católica Argentina, 2015. Disponible en: http://bibliotecadigital.uca.edu.ar/repositorio/tesis/metodologias-desarrollo-software.pdf. 19-03-2023)

Tipos de metodologías clásicas

Este tipo de metodologías de desarrollo de software se caracterizan por definir total y rigurosamente los requisitos al inicio de cada proyecto de ingeniería de software. Los ciclos en los cuáles se desarrolla el software, son poco flexibles y no permiten realizar cambios, entre los que consideramos:

Clásico, lineal o en cascada

1.1 Cascada en V

1.2 Cascada en fases solapadas

1.3 Cascada con subproyectos

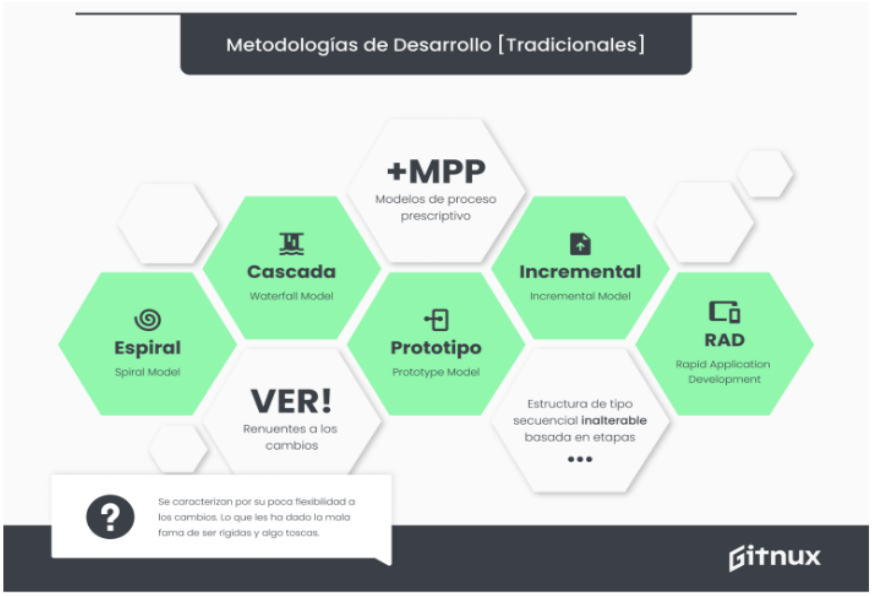
1.4 Cascada con reducción de riesgos

Basado en prototipos

Desarrollo rápido de aplicaciones (RAD)

Figura 9

Metodologías de Desarrollo Tradicional



Nota. En la figura "Metodologías de Desarrollo Tradicional" se detallan los diferentes tipos de metodologías tradicionales. Fuente: https://blog.gitnux.com/es/metodologias-de-desarrollo-de-software/

Clásico, lineal o en cascada. – Su versión original se debe a W. W. Royce [Royce, 1970], apareciendo después numerosos refinamientos. Este modelo sigue una secuencia lineal y secuencial de fases que incluyen el análisis de requisitos, el diseño, la implementación, las pruebas y el mantenimiento. Cada fase debe completarse antes de pasar a la siguiente. Esta metodología concibe el trabajo en un conjunto de etapas que deben ejecutarse una tras otra, su nombre se debe a las diferentes fases que componen el proyecto, ya que deben colocarse una sobre otra siguiendo un orden concreto y estricto de arriba hacia abajo. No podemos, por ejemplo, empezar la fase de diseño sin haber terminado la fase de requisitos.

Cascada impulsa la filosofía paso a paso, por bloques de tareas. (Sommerville,2005).

El modelo se divide en varias fases secuenciales que se llevan a cabo una después de la otra. A continuación, se describen las fases del modelo en cascada:

- *Requisitos:* En esta fase, se identifican y documentan los requisitos del software. Esto implica reunirse con los clientes para comprender sus necesidades y expectativas, y luego escribir una especificación de requisitos que describa detalladamente lo que el software debe hacer.
- *Diseño:* En esta fase, se desarrolla una solución de diseño para el software en función de los requisitos establecidos en la fase anterior. Esto implica la creación de diagramas y modelos que describan la arquitectura del software, la interfaz de usuario, la lógica de negocio y la base de datos.
- *Implementación:* En esta fase, se lleva a cabo la codificación del software utilizando el diseño desarrollado en la fase anterior. El código se escribe en un lenguaje de programación y se prueba para asegurarse de que funciona como se espera.

Figura 10

Fases del Modelo en Cascada

Nota. En la figura "Fases del Modelo en Cascada" se detallan las fases del modelo en cascada. Fuente: http://ingenieriaabril.blogspot.com/2016/09/metodologias-para-el-desarrollo-de.html.

Pruebas: En esta fase, se realiza una serie de pruebas para asegurarse de que el software cumple con los requisitos establecidos en la fase de requisitos. Estas pruebas pueden incluir pruebas unitarias, de integración y de aceptación.

Mantenimiento: En esta fase, se realiza el mantenimiento del software para corregir errores, agregar nuevas funciones y actualizar el software para que siga siendo relevante con el tiempo. Esto puede incluir la solución de problemas y la corrección de errores.

El modelo en cascada es un enfoque estructurado y ordenado para el desarrollo de software, pero puede ser inflexible si se encuentran problemas durante el proceso de desarrollo. Como habíamos mencionado anteriormente, muchas veces, las fases no pueden volver a estructurarse una vez que se han completado, lo que hace que el proceso de corrección de errores sea más difícil.

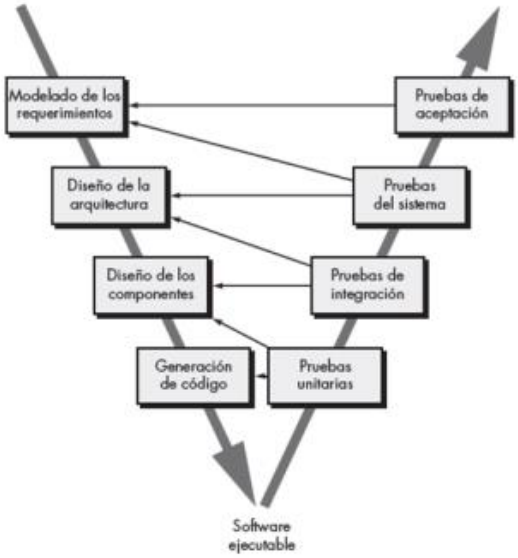
Modificaciones al modelo en cascada

Con el fin de solucionar las restricciones del modelo en cascada, surgieron varios modelos que lo transforman de alguna manera. Presentamos los principales modelos que surgieron a partir de hacer alguna modificación al modelo original de la metodología en cascada.

- *Cascada en V.-* Se remonta a los años 70 y fue creada como una variante de desarrollo posterior al modelo de cascada, donde se considera la relación entre las actividades para el aseguramiento de la calidad y aquellas asociadas con la comunicación, modelado y construcción temprana. En la cascada en V cada fase del desarrollo tiene una fase correspondiente de verificación y validación.

Figura 11

Modelo en Cascada en V



Nota. En la figura "Modelo en Cascada en V" se detalla el proceso que se realiza de forma secuencial hacia abajo y ejecuta una serie de pruebas. Fuente: https://ingenieriaensoftwarenathalyalava.wordpress.com/2015/04/25/modelos-de-procesos-prescriptivos/

Ventajas:

- La relación entre las etapas de desarrollo y los tipos de pruebas facilitan la localización de fallos.
- Es un modelo sencillo y de fácil aprendizaje.
- Hace explícito parte de la iteración y trabajo que hay que revisar.
- Especifica bien los roles de los distintos tipos de pruebas a realizar.
- Involucra al usuario en las pruebas.

Desventajas:

- Es difícil que el cliente exponga explícitamente todos los requisitos.
- El cliente debe tener paciencia pues obtendrá el producto al final del ciclo de vida.
- Las pruebas pueden llegar a tener un alto valor económico y, a veces, no lo suficientemente efectivas.
- El producto final conseguido puede que no refleje todos los requisitos del usuario.

Cascada en fases solapadas. – Para Peter Degrace, el modelo en cascada conocido también como "Modelo Sashimi", por su semejanza al estilo de presentación en rodajas de pescado crudo en Japón, es un ciclo en cascada, en el cual las etapas tienen un solapamiento, lo que favorece al avance del proyecto, permitiendo avanzar de una etapa hacia otra, sin concluir completamente dicha etapa.

Figura 12

Modelo Cascada en Fases Solapadas

Nota. En la figura "Modelo Cascada en fases solapadas" se detalla las fases en el modelado en cascada con fases solapadas. Fuente: McConnell, 1997.

Ventajas:

- Una etapa retroalimenta a la anterior y por consiguiente se la puede mejorar.
- No es necesario una documentación detallada en cada etapa ya que estas se comparten.
- Su planificación es más sencilla.
- Permite la optimización de los recursos de CPU, memoria, espacio, si la aplicación se comparte con otros sistemas.
- Se obtiene mayor eficacia y calidad en el producto final.

Desventajas:

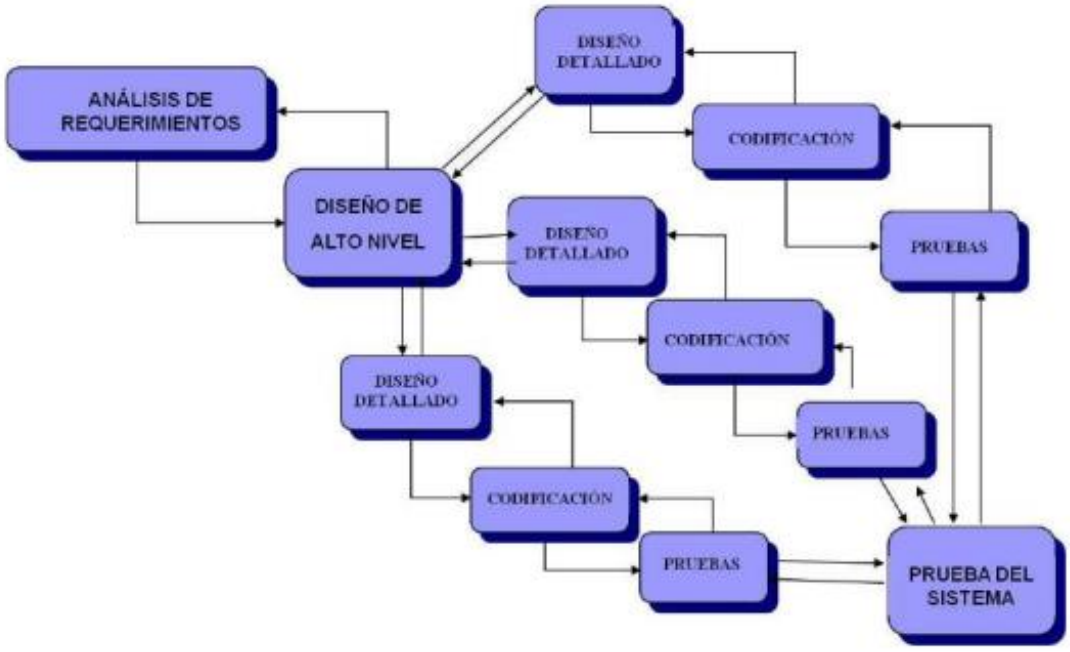
- Es difícil gestionar el comienzo y fin de cada etapa.

- Difícil controlar el progreso del proyecto debido a que los finales de fase ya no son un punto de referencia claro.
- Dificultad para reconocer todos los requerimientos desde el inicio.
- El producto final conseguido puede que no refleje todos los requisitos del usuario.

Cascada con subproyectos. – Se origina en la década de 1960, teniendo como fin desarrollar un sistema enfocado en los negocios de una manera eficiente, en una época donde la gran demanda la abarcaban los conglomerados empresariales. La idea al inicio de su concepción era realizarla de manera estructurada, reiterando cada una de las etapas del ciclo de vida del software, para luego llegar a su demostración como un ciclo que permite la ejecución de algunas de las tareas en paralelo.

Figura 13

Modelo Cascada con Subproyectos



Nota. En la figura "Modelo Cascada con subproyectos" En el modelo de cascada con subproyectos se permite la ejecución de ciertas tareas de la cascada, pero en paralelo (subproyectos). Fuente: Fuentes et al, 2019

Ventajas:

- Se desarrolla más rápidamente.

- Se puede separar en subproyectos y cada uno puede proceder a su forma y su ritmo.
- Su planificación es sencilla.
- La calidad del producto resultante es alta.
- Permite trabajar con personal poco cualificado.

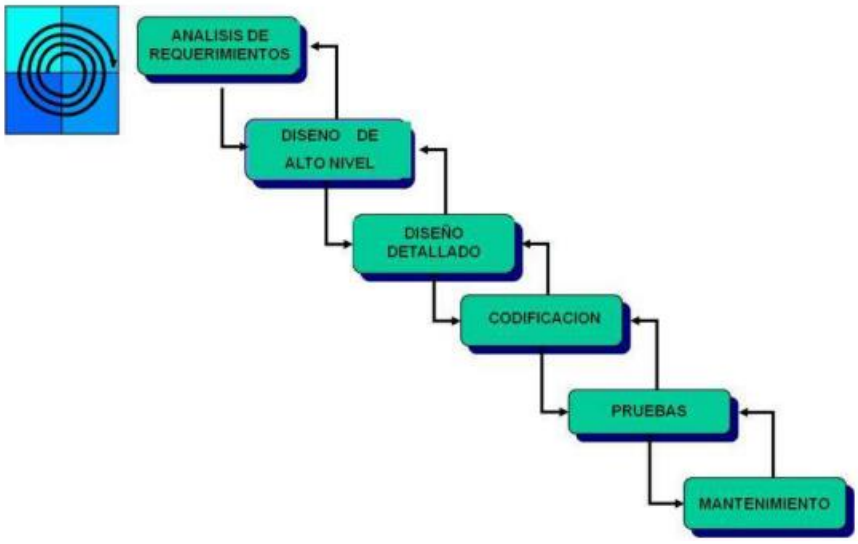
Desventajas:

- Si se han cometido errores en una fase es difícil volver atrás.
- El cliente no verá resultados hasta el final.
- No tiene buena aceptación de cambios en los requerimientos.
- Puede existir interdependencias no previstas entre los subproyectos.

Cascada con reducción de riesgos. - Este modelo, concentra una espiral en lo alto de la cascada para controlar el riesgo y desarrollo de los requerimientos. En este nivel es posible desarrollar un prototipo de interfaz de usuario, manejar apuntes, realizar entrevistas a los usuarios, observar cómo los usuarios interactúan con algún sistema más antiguo, o utilizar otros métodos que se consideren apropiados para poder identificar los requerimientos.

Figura 14

Modelo Cascada con reducción de riesgos



Nota. En la figura "Modelo Cascada con reducción de riesgos" En este modelo observamos que se incorpora una espiral en lo alto de la cascada. Fuente: Fuentes et al, 2019.

Ventajas:

- Reduce el riesgo de producir un sistema que no cumpla con los requerimientos.
- Se realiza documentación completa del sistema.
- Permite iterar sobre las etapas iniciales del proyecto.
- Presenta robustez en cuanto a las funcionalidades del sistema.
- Acertamiento en la determinación de los roles que se manejan.

Desventajas:

- Si se plantean mal los requerimientos, esto sólo se reflejará al final.
- Las modificaciones realizadas en los requerimientos afectarán significativamente una vez que se esté trabajando en etapas avanzadas.

Tabla 6

Resumen de las características de modelos derivados del modelo en cascada

Modelo	Características
Cascada en V	Cada fase del desarrollo tiene una fase correspondiente de verificación y validación
Cascada con fases solapadas	Permite avanzar a la siguiente etapa antes de terminar la actual.
Cascada con subproyectos	Permite la ejecución de algunas de las tareas de la cascada en paralelo (subproyectos), siempre que se haya realizado una cuidadosa planificación.
Cascada con reducción de riesgos	Incorpora una espiral en lo alto de la cascada para controlar el riesgo de los requerimientos: se hacen prototipos de interfaz de usuario, entrevistas, etc.

Nota. En la tabla se hace un resumen de las principales características de los modelos de desarrollo de software derivados del modelo en cascada. Fuente: Autores

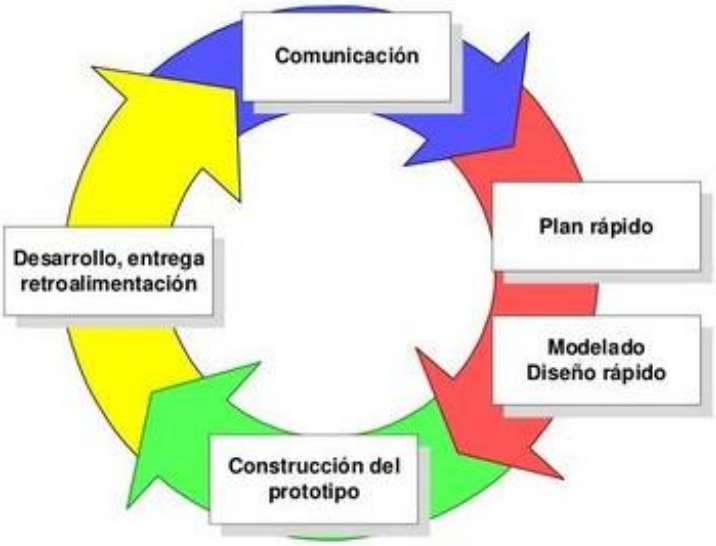
Basado en prototipos. – Fue en la década de 1970 cuando se popularizó la técnica de desarrollo de software basado en prototipos, con el trabajo del ingeniero de software Barry Boehm y su equipo en la Universidad de California, Los Ángeles.

Boehm desarrolló el enfoque de prototipos para el desarrollo de software, en respuesta a la necesidad de desarrollar sistemas complejos y grandes en un entorno de incertidumbre y cambio constante. La idea era que la creación de prototipos funcionales permitiría a los desarrolladores obtener retroalimentación temprana del usuario y ajustar rápidamente el diseño, lo que reduciría el riesgo de desarrollar un sistema que no cumpliera con las necesidades del usuario.

Desde entonces, el desarrollo de software basado en prototipos se ha convertido en una técnica ampliamente utilizada en la industria del software, y el modelado basado en prototipos se ha convertido en una técnica popular para la creación de modelos iterativos de software. Al utilizar esta técnica, los desarrolladores pueden crear prototipos rápidos de modelos de software y ajustarlos en base a los comentarios de los usuarios, lo que les permite refinar rápidamente el diseño antes de construir la versión final del software.

Figura 15

Modelo basado en prototipos



Nota. En la figura "Modelo basado en prototipos" podemos observar las fases de este tipo de modelo. Fuente:
https://www.goconqr.com/diapositiva/3204595/modelos-basados-en-protitipos

El desarrollo de software basado en prototipos es un enfoque de desarrollo de software que se centra en la creación rápida de prototipos de software funcional para obtener retroalimentación de los usuarios y refinar el diseño antes de construir la versión final del software.

El modelado basado en prototipos es una técnica específica de este enfoque, que se utiliza para crear modelos de software iterativos y ajustados en base a los comentarios de los usuarios. La idea es que, al crear prototipos funcionales del software, los desarrolladores pueden obtener comentarios de los usuarios y realizar ajustes rápidos en el diseño antes de construir la versión final del software.

Ventajas:

- El prototipo se desarrolla en menos tiempo para poder ser probado o testeado.
- El prototipo debe incluir los requisitos y características básicas de la aplicación para poder evaluar su funcionamiento y utilidad.
- Evoluciona gracias a la interacción con los usuarios.
- Tienen un costo bajo de desarrollo.

Desventajas:

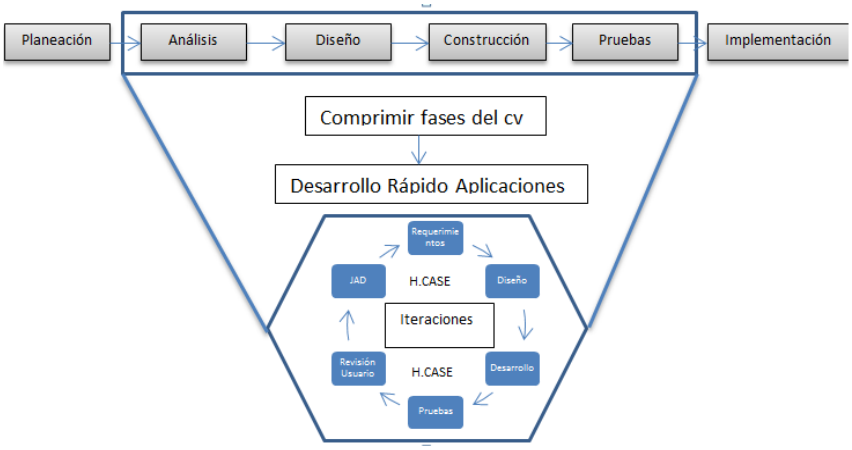
- Se suelen desatender aspectos importantes, tales como la calidad y el mantenimiento a largo plazo.
- El desarrollador suele tomar algunas decisiones de implementación poco convenientes.

Desarrollo rápido de aplicaciones (RAD).- (por sus siglas en inglés Rapid Application Development o Desarrollo Rápido de Aplicaciones). Es un enfoque de desarrollo de software que se enfoca en la rapidez y la eficiencia en la creación de aplicaciones. RAD se originó en la década de 1980, pero presentado oficialmente por James Martin en 1991, como una respuesta a la necesidad de desarrollar software más rápidamente, en un entorno de negocios que cambia rápidamente y donde los plazos de entrega son cada vez más cortos.

El enfoque de RAD se centra en la utilización de técnicas de prototipado, iteración y reutilización de software para acelerar el proceso de desarrollo de aplicaciones. En lugar de seguir un enfoque tradicional de desarrollo de software que se enfoca en la recopilación exhaustiva de requisitos, RAD se enfoca en el rápido desarrollo de prototipos funcionales y en la retroalimentación temprana del usuario para refinar el diseño.

Figura 16

Metodología R.A.D



Nota. En la figura "Metodología RAD" El enfoque RAD se ha convertido en una técnica popular en la industria del software, y se utiliza en una amplia gama de aplicaciones. Fuente: https://www.goconqr.com/diapositiva/3204595/modelos-basados-en-protitipos

El enfoque RAD se basa en los siguientes principios:

- *Prototipado rápido:* RAD se enfoca en la creación rápida de prototipos funcionales, que permiten a los desarrolladores obtener retroalimentación temprana del usuario y refinar el diseño de manera iterativa.
- *Reutilización de software:* RAD se enfoca en la reutilización de componentes de software existentes, lo que acelera el proceso de desarrollo y mejora la calidad del software.
- *Colaboración entre desarrolladores y usuarios:* RAD se enfoca en la colaboración entre desarrolladores y usuarios para asegurar que las necesidades del usuario se cumplan de manera efectiva.
- *Ciclos de desarrollo cortos:* RAD se enfoca en la entrega de software en ciclos de desarrollo cortos, lo que permite a los usuarios obtener valor de manera temprana y mejora la retroalimentación del usuario para refinar el diseño.

Cada modelo de desarrollo de software tiene sus propias ventajas y desventajas, y la elección del modelo dependerá de las necesidades y requisitos específicos del proyecto y del equipo de desarrollo.

3.1.2 MODELOS EVOLUTIVOS

Los modelos evolutivos pueden ser utilizados para entender cómo cambian las características de los sistemas a lo largo del tiempo. Estos modelos pueden ser aplicados a distintos tipos de sistemas, como por ejemplo para poblaciones, sistemas sociales, económicos y tecnológicos.

Por ejemplo, un modelo evolutivo podría ser utilizado para estudiar cómo cambian las características de un sistema de transporte a medida que la tecnología y las preferencias de los usuarios evolucionan. El modelo podría incluir factores como la demanda de transporte, el costo y la eficiencia de diferentes modos de transporte, y la capacidad de adaptación del sistema a nuevos avances tecnológicos.

Otro ejemplo sería aplicar modelos evolutivos al análisis de sistemas de energía renovable, para entender cómo las tecnologías solares, eólicas u otras pueden evolucionar para mejorar su eficiencia y reducir su costo. El modelo podría considerar la competencia entre diferentes tecnologías y las condiciones del mercado, así como la influencia de las políticas gubernamentales y las preocupaciones ambientales.

Los modelos evolutivos son considerados como una herramienta poderosa para el análisis de sistemas, permitiendo una comprensión más profunda de cómo cambian las características de los sistemas a lo largo del tiempo y cómo pueden ser mejorados y adaptados para satisfacer las necesidades cambiantes de la sociedad.

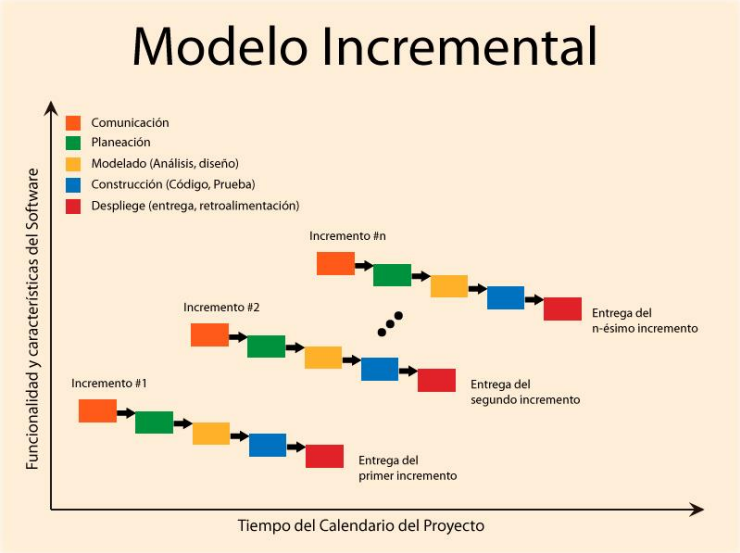
En el análisis de sistemas, existen varios tipos de modelos evolutivos que pueden ser utilizados para estudiar cómo cambian las características de los sistemas a lo largo del tiempo. Algunos de los tipos más comunes son:

- Incremental
- Iterativo
- En espiral

Modelo Evolutivo Incremental. - El modelo incremental es un enfoque utilizado en el análisis de sistemas para desarrollar un sistema en etapas iterativas y progresivas. En lugar de intentar desarrollar el sistema completo en una sola vez, el modelo incremental implica construir el sistema en pequeñas partes o módulos, añadiendo gradualmente nuevas funcionalidades en cada iteración.

Figura 17

Esquema del modelo incremental



Nota. En la figura "Esquema del modelo incremental", observamos el incremento de las etapas en función del tiempo. Fuente: http://isw-udistrital.blogspot.com/2012/09/ingenieria-de-software-i.html

El proceso del modelo incremental generalmente implica los siguientes pasos:

- *Identificación de los requisitos iniciales* del sistema y selección de los módulos o componentes más críticos que deben ser desarrollados primero.
- *Desarrollo de los módulos iniciales*, con un enfoque en la funcionalidad crítica y una arquitectura flexible que permita la adición de nuevos módulos en el futuro.
- *Pruebas y verificación de los módulos iniciales* para asegurarse de que funcionan correctamente y cumplen con los requisitos.
- *Agregar gradualmente nuevos módulos y funcionalidades*, probando y verificando cada iteración antes de avanzar a la siguiente.

El modelo incremental tiene varias ventajas. En primer lugar, permite una mayor flexibilidad en el proceso de desarrollo, lo que significa que los cambios y las adiciones pueden ser más fáciles de realizar a medida que el sistema evoluciona. En segundo lugar, permite una mayor eficiencia en el uso de los recursos, ya que los recursos se pueden concentrar en los módulos más críticos en primer lugar, y se pueden hacer ajustes y mejoras a medida que se agregan nuevos módulos. Por último, el modelo incremental puede ser menos riesgoso, ya que los problemas y las limitaciones del sistema pueden ser identificados y abordados más temprano en el proceso.

Sin embargo, el modelo incremental también tiene algunas limitaciones. En particular, puede ser difícil predecir el resultado final del sistema a medida que se agregan nuevos módulos, lo que puede dificultar la planificación y la estimación del tiempo y los recursos necesarios para completar el sistema. Además, la integración de los diferentes módulos puede ser complicada, especialmente si se ha producido un cambio significativo en los requisitos o la arquitectura del sistema a medida que se ha desarrollado.

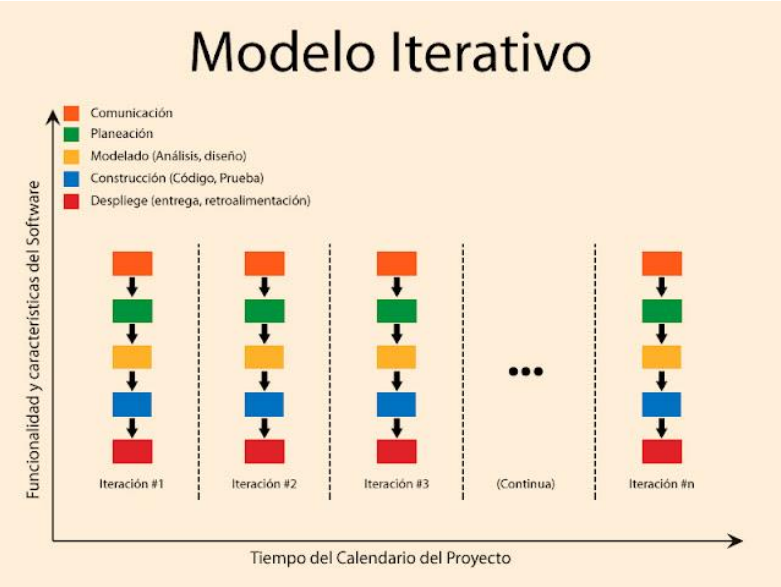
Modelo Evolutivo Iterativo. – es un enfoque utilizado en el análisis de sistemas que se centra en el desarrollo iterativo y evolutivo de un sistema, en lugar de en una planificación detallada del proyecto al principio del proceso. Este modelo se basa en la idea de que los requisitos del sistema pueden cambiar con el tiempo, y que el proceso de desarrollo debe adaptarse para incorporar estos cambios.

El modelo evolutivo iterativo se desarrolla en varias fases iterativas. Cada fase incluye la planificación, el análisis, el diseño, la implementación y la evaluación, y cada fase se enfoca en una parte específica del sistema.

En cada iteración, se identifican los requisitos adicionales del sistema y se agregan a la solución actual. Esto se logra mediante el uso de prototipos, que son modelos simplificados del sistema que se crean rápidamente para probar nuevas ideas y funcionalidades.

Figura 18

Esquema del modelo evolutivo iterativo



Nota. En la figura "Esquema del modelo Iterativo", observamos el desarrollo de las etapas en función del tiempo. Fuente: http://isw-udistrital.blogspot.com/2012/09/ingenieria-de-software-continuacion.html

El proceso de iteración permite que el sistema evolucione a medida que se desarrolla, en lugar de tratar de construir el sistema completo de una sola vez. Esto proporciona una mayor flexibilidad y permite que el equipo de desarrollo se adapte a los cambios en los requisitos del sistema a lo largo del tiempo.

El modelo evolutivo iterativo es útil en situaciones donde los requisitos del sistema no están completamente definidos desde el principio. También es útil cuando se espera que los requisitos cambien con el tiempo, o cuando se necesita una solución rápida y provisional para una parte específica del sistema.

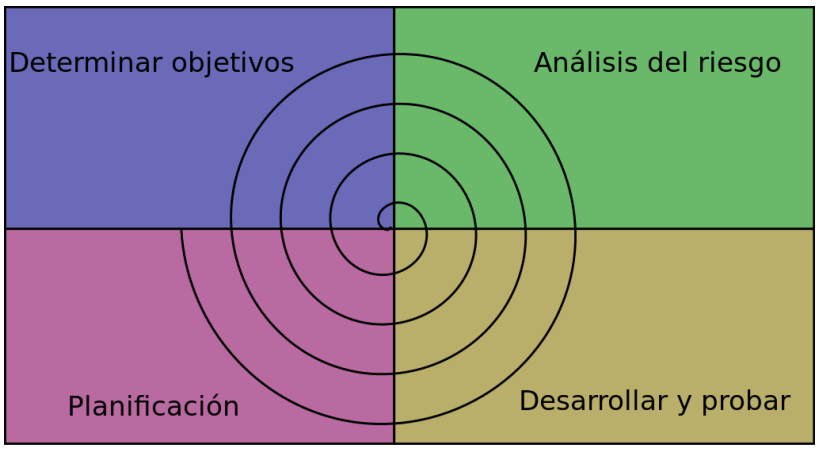
Sin embargo, el modelo evolutivo iterativo también puede tener algunas limitaciones. Puede ser difícil estimar el tiempo y el costo del proyecto, ya que el proceso de desarrollo es más fluido y menos predecible que en otros modelos. Además, el uso de prototipos puede requerir una inversión significativa de tiempo y recursos, lo que puede afectar el presupuesto del proyecto.

Modelo Evolutivo Espiral. - es un enfoque en la ingeniería de software que se utiliza para el análisis de sistemas. Este modelo se basa en la iteración y la retroalimentación continua a lo largo de todo el ciclo de vida del software, lo que permite a los desarrolladores y a los usuarios finales trabajar juntos para mejorar y refinar el sistema.

El modelo evolutivo espiral consta de cuatro fases principales: planificación, análisis de riesgos, ingeniería y evaluación. En la fase de planificación, se identifican los objetivos del sistema y se establecen los planes para alcanzarlos. En la fase de análisis de riesgos, se identifican y evalúan los posibles riesgos y se establecen estrategias para mitigarlos, como se observa en la figura 19.

Figura 19

Esquema del modelo espiral



Nota. En la figura "Esquema del modelo espiral", observamos el desarrollo de las etapas en el desarrollo del modelo espiral.
Fuente:
https://es.wikipedia.org/wiki/Desarrollo_en_espiral#/media/Archivo:ModeloEspiral.svg

En la fase de ingeniería, se desarrolla el software de acuerdo con los requisitos establecidos en las fases anteriores. Y en la fase de evaluación, se realiza una revisión del sistema completo y se evalúa su funcionamiento para determinar si cumple con los requisitos establecidos y para identificar posibles mejoras. El modelo evolutivo espiral es un enfoque iterativo y flexible para el análisis y desarrollo de sistemas de software. Las fases del modelo evolutivo espiral de manera más detallada se resumen a continuación:

- ***Planificación:*** En esta fase, se establecen los objetivos y los requisitos del sistema, se identifican los recursos necesarios y se desarrolla un plan general para el proyecto. También se establece un calendario para el desarrollo del sistema.
- ***Análisis de riesgos:*** En esta fase, se identifican los riesgos asociados con el proyecto y se establecen estrategias para mitigarlos. También se realiza un análisis de viabilidad del proyecto.
- ***Ingeniería:*** En esta fase, se lleva a cabo la implementación del sistema de software de acuerdo con los requisitos establecidos en las fases anteriores. Esto incluye la programación, la codificación, la integración y las pruebas.
- ***Evaluación:*** En esta fase, se realiza una revisión completa del sistema y se evalúa su funcionamiento. También se identifican áreas para mejoras y se establecen planes para futuras iteraciones del proceso.

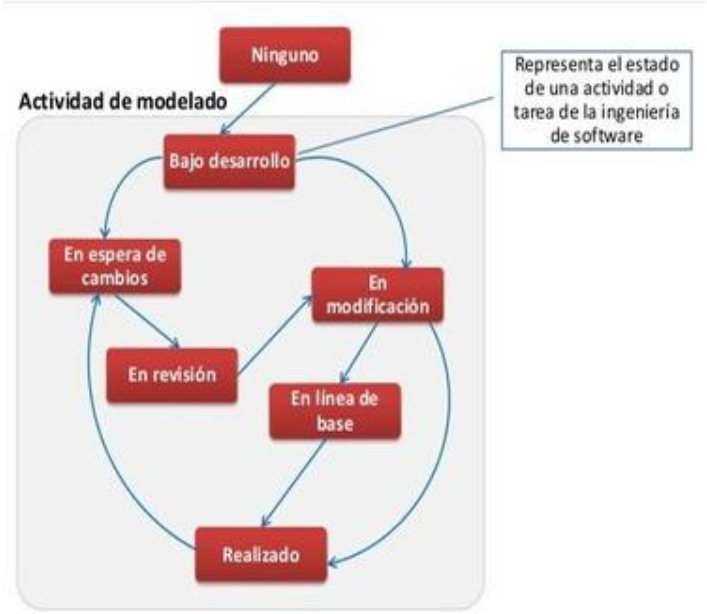
Cada una de estas fases puede incluir sub-fases adicionales para refinar el análisis, el diseño y la implementación del sistema. El modelo evolutivo espiral se caracteriza por su enfoque en la iteración continua y la retroalimentación para mejorar el sistema de software a lo largo del tiempo.

El modelo evolutivo espiral es especialmente útil para proyectos grandes y complejos, donde los requisitos pueden cambiar a lo largo del tiempo y es necesario tener una mayor flexibilidad en el proceso de desarrollo. Además, el enfoque en la identificación y mitigación de riesgos puede ayudar a reducir los problemas y retrasos en el proyecto.

Modelo evolutivo concurrente. - se centra en la colaboración simultánea de múltiples equipos en el análisis y desarrollo de sistemas de software complejos. Este modelo se basa en la premisa de que el desarrollo de software es una actividad compleja y que requiere la colaboración de múltiples equipos con diferentes habilidades y conocimientos.

Figura 20

Esquema modelo concurrente



Nota. En la figura "Esquema del modelo concurrente", observamos el desarrollo de las etapas en el desarrollo del modelo concurrente. Fuente: https://williantab.weebly.com/concurrente.html

En el modelo de desarrollo concurrente, se divide el proyecto en múltiples subproyectos y se asigna a cada uno de ellos un equipo de desarrollo independiente. Cada equipo

trabaja en su propia área de especialización, pero está en constante comunicación y colaboración con los otros equipos. El modelo evolutivo concurrente incluye las siguientes fases:

- ***Planificación:*** En esta fase se define el alcance del proyecto y se establecen los objetivos y las metas a largo plazo. También se define el equipo de desarrollo y se establecen los roles y las responsabilidades de cada miembro del equipo.
- ***Análisis:*** En esta fase, se lleva a cabo la definición de los requisitos y se establece la arquitectura general del sistema. Cada equipo de desarrollo trabaja en su área de especialización, pero está en constante comunicación con los otros equipos para asegurar que las partes individuales del sistema se integren correctamente.
- ***Diseño:*** En esta fase, se realiza el diseño detallado de cada uno de los componentes del sistema. Cada equipo de desarrollo trabaja en su área de especialización, pero de nuevo está en constante comunicación con los otros equipos para asegurar la coherencia y la integridad del sistema completo.
- ***Implementación:*** En esta fase, se lleva a cabo la programación y la codificación de cada uno de los componentes del sistema. Cada equipo de desarrollo trabaja en su área de especialización, pero está en constante comunicación con los otros equipos para asegurar que los componentes individuales se integren correctamente.
- ***Pruebas:*** En esta fase, se lleva a cabo la validación y verificación del sistema completo. Se realizan pruebas en todo el sistema para asegurarse de que funciona correctamente y cumple con los requisitos definidos.

El modelo de desarrollo concurrente es especialmente útil para proyectos de software grandes y complejos, donde múltiples equipos de desarrollo son necesarios para llevar a cabo el proyecto. La colaboración constante entre los equipos de desarrollo permite una mayor eficiencia en el proceso de desarrollo y puede reducir los errores y las incompatibilidades entre los diferentes componentes del sistema.

3.1.3 MODELOS ÁGILES

La metodología ágil es un enfoque iterativo e incremental para el análisis y desarrollo de sistemas de software que se basa en la colaboración estrecha entre el equipo de desarrollo y el cliente. El objetivo principal es asegurar que el producto final cumpla con los requisitos y necesidades del cliente. En lugar de una entrega única al final del

proyecto, la metodología ágil se enfoca en la entrega continua de software funcional en pequeñas iteraciones llamadas "sprints". Cada sprint se centra en la entrega de un conjunto específico de características o funcionalidades que se han definido previamente. Las fases de los modelos ágiles se representan de manera gráfica, como se muestra en la siguiente figura;

Figura 21

Esquema de las metodologías agiles

Nota. En la figura "Esquema de las metodologías ágiles", observamos el desarrollo de las etapas en la implementación de metodologías ágiles.
Fuente: https://www.progressalean.com/metodologia-agile/

Las metodologías ágiles más comunes incluyen Scrum y Kanban. Estas metodologías tienen las siguientes fases:

- *Planificación:* Se define el alcance del proyecto y se establecen los objetivos a largo plazo. También se establece el equipo de desarrollo y se asignan las responsabilidades de cada miembro.
- *Análisis:* Se definen los requisitos de usuario y se establece el conjunto de características o funcionalidades que se entregarán en el primer sprint.
- *Desarrollo:* Se lleva a cabo la implementación del software de acuerdo con los requisitos establecidos en el sprint anterior. El equipo de desarrollo trabaja en estrecha colaboración con el cliente para asegurarse de que el software cumpla con sus necesidades.

- *Pruebas:* Se llevan a cabo pruebas de calidad para asegurarse de que el software funciona correctamente y cumple con los requisitos establecidos.
- *Revisión:* Se realiza una revisión del software entregado en el sprint y se proporciona retroalimentación para mejorar el proceso en el siguiente sprint.

Las metodologías ágiles tienen varias ventajas, como la entrega continua de software funcional, la capacidad de adaptarse a los cambios en los requisitos del cliente y una mayor colaboración entre el equipo de desarrollo y el cliente. Sin embargo, también tienen limitaciones, como la necesidad de una comunicación constante y efectiva entre el equipo de desarrollo y el cliente y la falta de un diseño detallado previo, lo que puede llevar a problemas de integración y escalabilidad en sistemas más grandes y complejos.

Existen diferentes tipos de metodologías ágiles que se utilizan para el análisis y desarrollo de sistemas de software. A continuación, se describen algunas de las metodologías ágiles más útiles:

- Scrum
- Kanban
- Extreme Programming (XP)
- Lean Software Development
- Crystal
- Adaptive Software Development

Scrum. - es una de las metodologías ágiles más populares para el análisis y desarrollo de sistemas de software. Se enfoca en la entrega continua de software funcional en pequeñas iteraciones llamadas "sprints", lo que permite una mayor adaptabilidad a los cambios en los requisitos del cliente.

En el modelo ágil Scrum, se divide el proyecto en sprints de dos a cuatro semanas de duración. Cada sprint se centra en la entrega de un conjunto específico de características o funcionalidades que se han definido previamente. Durante cada sprint, el equipo de desarrollo trabaja en estrecha colaboración con el cliente para asegurarse de que el software cumpla con sus necesidades. Como podemos observar en la siguiente figura (22)

se detallan las fases;

Figura 22

Esquema de los modelos Scrum

Nota. En la figura "Esquema de la metodología scrum", observamos el desarrollo de las etapas en la implementación de la metodología scrum.
Fuente: https://blog.wearedrew.co/productividad/-ventajas-y-desventajas-de-la-metodologia-scrum

Scrum tiene las siguientes fases:

- *Planificación del sprint:* Se define el conjunto de características o funcionalidades que se entregarán en el sprint y se establece el objetivo del sprint. También se define el equipo de desarrollo y se asignan las tareas específicas a cada miembro.
- *Reuniones diarias:* El equipo de desarrollo se reúne diariamente para revisar el progreso y asegurarse de que se están cumpliendo los objetivos del sprint.
- *Revisión del sprint:* Al final de cada sprint, se realiza una revisión del software entregado y se proporciona retroalimentación para mejorar el proceso en el siguiente sprint.
- *Retrospectiva del sprint:* También al final de cada sprint, se lleva a cabo una retrospectiva para analizar lo que funcionó bien y lo que se puede mejorar en el proceso.

Scrum tiene varias ventajas, como la entrega continua de software funcional, una mayor adaptabilidad a los cambios en los requisitos del cliente y una mayor colaboración entre

el equipo de desarrollo y el cliente. Sin embargo, también tiene limitaciones, como la necesidad de una comunicación constante y efectiva entre el equipo de desarrollo y el cliente y la falta de un diseño detallado previo, lo que puede llevar a problemas de integración y escalabilidad en sistemas más grandes y complejos.

Kanban. - es un enfoque ágil de gestión de proyectos que se centra en la entrega constante y gradual de un producto o servicio. En el análisis de sistemas, Kanban puede utilizarse para mejorar la eficiencia y la productividad del equipo de análisis de sistemas, permitiendo una mayor flexibilidad y adaptación a los cambios en los requisitos del proyecto. Se basa en el uso de tableros visuales, donde se muestran las tareas pendientes, en progreso y completadas. Cada tarea se representa por una tarjeta, que contiene información sobre la tarea, como su nombre, descripción, prioridad y estado actual.

En el análisis de sistemas, las tarjetas pueden representar los requisitos del sistema o las funcionalidades que deben ser desarrolladas. El equipo de análisis de sistemas puede utilizar el tablero Kanban para visualizar el progreso del proyecto y asignar tareas a los miembros del equipo, como se muestra en la siguiente figura.

Figura 23

Esquema del modelo Kanban

Nota. En la figura "Esquema de la metodología Kanban", observamos el desarrollo de las actividades en la implementación de la metodología Kanban.
Fuente: https://tecnosoluciones.com/10-razones-para-usar-la-metodologia-kanban-en-tu-organizacion/

Además, la metodología Kanban permite una mayor flexibilidad en el proceso de desarrollo, ya que los cambios en los requisitos pueden ser incorporados fácilmente en el proceso, sin afectar el progreso del proyecto. Esto se logra al permitir que las nuevas tareas se agreguen al tablero Kanban en cualquier momento, sin afectar las tareas en curso.

Otro beneficio de la metodología Kanban en el análisis de sistemas es que promueve la colaboración y la comunicación dentro del equipo. Al visualizar el estado actual del proyecto, los miembros del equipo pueden identificar rápidamente las áreas que requieren atención y trabajar juntos para resolver los problemas. La metodología Kanban puede ser una herramienta muy útil en el análisis de sistemas, ya que permite una gestión eficiente y flexible del proyecto, mejorando la productividad y la colaboración del equipo.

Extreme Programming (XP). - es un enfoque ágil de desarrollo de software que se centra en la entrega rápida y continua de software de alta calidad. Fue desarrollada en la década de 1990 por Kent Beck y ha ganado popularidad en los últimos años debido a su enfoque pragmático y eficiente. Este tipo de metodología se basa en cuatro valores fundamentales: comunicación, simplicidad, retroalimentación y coraje. Estos valores se aplican en todas las etapas del proceso de desarrollo de software y se utilizan para guiar la toma de decisiones y la planificación del proyecto como se muestra en la figura.

Figura 24

Esquema del Extreme Programming

Nota. En la figura "Esquema de la metodología XP", observamos los valores más importantes para desarrollar este tipo de metodología. Fuente: http://www.pmoinformatica.com/2012/11/los-5-valores-de-la-programacion.html

XP se enfoca en la colaboración cercana entre los miembros del equipo de desarrollo, incluyendo a los clientes, para asegurar que se entregue el software correcto. Además, XP se centra en la calidad del software y en la entrega rápida de versiones funcionales del software a los clientes. Utiliza varias prácticas para lograr estos objetivos, incluyendo la planificación del juego (planning game), la integración continua (continuous integration), pruebas unitarias (unit testing), refactorización (refactoring), diseño

simple (simple design), programación en parejas (pair programming) y desarrollo guiado por pruebas (test-driven development).

La metodología XP se enfoca en la entrega rápida y continua de software de alta calidad a través de la colaboración cercana entre los miembros del equipo de desarrollo y los clientes. Utiliza una variedad de prácticas para lograr estos objetivos, incluyendo la integración continua, pruebas unitarias y programación en parejas.

Lean Software Development. - es una metodología ágil de desarrollo de software que se centra en la creación de valor para el cliente a través de la eliminación de desperdicios y la mejora continua del proceso de desarrollo. En el análisis de sistemas, Lean Software Development puede ayudar a los equipos a ser más eficientes y a entregar productos de mayor calidad.

La metodología Lean se basa en siete principios fundamentales: eliminar el desperdicio, amplificar el aprendizaje, tomar decisiones lo más tarde posible, entregar rápidamente, empoderar al equipo, construir la calidad en el proceso y ver el todo, como se puede ver en la siguiente figura.

Figura 25

Esquema de la metodología Lean Software Development

Nota. En la figura "Esquema de la metodología Lean Software Development", observamos los principios implementados en este tipo de metodología. Fuente: https://netmind.net/es/lean-software-development-lsd-los-siete-principios/

En el análisis de sistemas, se pueden aplicar estos principios de la siguiente manera:

- *Eliminar el desperdicio:* En el análisis de sistemas, el desperdicio puede manifestarse en la forma de requisitos innecesarios o mal definidos,

documentación excesiva, reuniones improductivas, entre otros. Los equipos pueden identificar y eliminar estos desperdicios para centrarse en lo que es importante para el cliente.

- *Amplificar el aprendizaje:* Los equipos de análisis de sistemas pueden mejorar continuamente su proceso mediante la revisión y reflexión sobre su trabajo y el intercambio de conocimientos con otros miembros del equipo.
- *Tomar decisiones lo más tarde posible:* Al posponer la toma de decisiones, los equipos pueden tener más información disponible para tomar decisiones más informadas y evitar costosas correcciones más adelante en el proceso de desarrollo.
- *Entregar rápidamente:* Los equipos pueden utilizar entregas frecuentes y pequeñas para obtener retroalimentación temprana del cliente y adaptar el desarrollo del software según sea necesario.
- *Empoderar al equipo:* Los equipos de análisis de sistemas pueden beneficiarse de una mayor autonomía para tomar decisiones y resolver problemas, lo que puede conducir a un mayor compromiso y motivación.
- *Construir la calidad en el proceso:* Los equipos pueden utilizar prácticas de prueba y revisión para garantizar que el software entregado sea de alta calidad.
- *Ver el todo:* Los equipos pueden adoptar una perspectiva amplia del proceso de desarrollo, incluyendo la comunicación con el cliente, para asegurarse de que están entregando un producto que cumple con las necesidades del cliente y agrega valor a su negocio.

Lean Software Development puede ser una metodología valiosa en el análisis de sistemas al ayudar a los equipos a centrarse en el valor para el cliente y mejorar continuamente su proceso de desarrollo.

Crystal. - se centran en la colaboración, la comunicación y el desarrollo iterativo e incremental. Se basa en la idea de que diferentes proyectos tienen diferentes necesidades, por lo que se deben adaptar las prácticas de la metodología a cada proyecto en particular.

En el análisis de sistemas, el modelo Crystal puede ser beneficioso al permitir que los equipos adapten su enfoque a las necesidades específicas del proyecto. A continuación, se presentan algunos aspectos clave del modelo Crystal que pueden aplicarse en el análisis de sistemas:

- *Tamaño del equipo:* El modelo Crystal reconoce que el tamaño del equipo afecta la forma en que se debe llevar a cabo el trabajo. Los proyectos más grandes pueden requerir más documentación y coordinación, mientras que los equipos más pequeños pueden permitir una mayor colaboración.
- *Comunicación:* La comunicación es fundamental en el modelo Crystal. Los equipos deben trabajar juntos de manera cercana y efectiva para asegurarse de que están en sintonía con los objetivos del proyecto y con el cliente. La comunicación efectiva también ayuda a prevenir problemas y a garantizar que el proyecto se mantenga en el camino correcto.
- *Desarrollo iterativo:* El desarrollo iterativo es un componente clave del modelo Crystal. Los equipos deben desarrollar el software en pequeñas iteraciones, cada una de las cuales debe estar enfocada en producir un conjunto de funcionalidades de alta calidad.
- *Roles flexibles:* El modelo Crystal reconoce que diferentes proyectos pueden requerir diferentes roles dentro del equipo. En el análisis de sistemas, esto puede significar que los miembros del equipo desempeñen múltiples roles para adaptarse a las necesidades del proyecto.
- *Adaptabilidad:* El modelo Crystal se basa en la idea de que los proyectos pueden cambiar a medida que avanza el tiempo. Por lo tanto, los equipos deben ser flexibles y estar dispuestos a cambiar sus prácticas y enfoques a medida que sea necesario para satisfacer las necesidades cambiantes del proyecto y del cliente.

Figura 26

Esquema Metodología Crystal

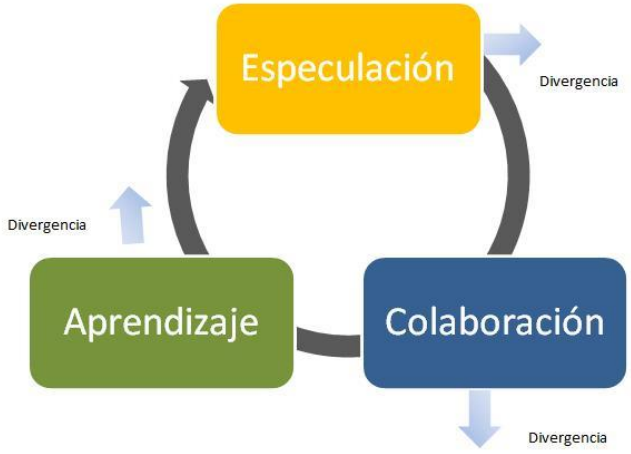
Nota. En la figura "Esquema de la metodología Crystal", observamos como se conforma el equipo de trabajo en este tipo de metodologías.
Fuente: https://www.timetoast.com/timelines/metodologia-crystal-clear

El modelo Crystal puede ser una metodología útil en el análisis de sistemas al permitir que los equipos adapten su enfoque a las necesidades específicas del proyecto. Los equipos pueden ajustar el tamaño del equipo, enfatizar la comunicación efectiva, adoptar un enfoque iterativo, desempeñar roles flexibles y ser adaptables a medida que se presentan nuevos desafíos y oportunidades.

Adaptive Software Development. - (ASD) es una metodología ágil de desarrollo de software que se enfoca en la adaptación continua del proceso de desarrollo para cumplir con las necesidades cambiantes del negocio y del usuario. En el análisis de sistemas, la metodología (ASD) se enfoca en la recopilación de requisitos, en el diseño y en la planificación del sistema de manera iterativa e incremental. Esta metodología se basa en el entendimiento de que los requisitos cambian y evolucionan a lo largo del tiempo, por lo que se debe estar preparado para adaptarse a esos cambios. Las fases en las que se desarrolla este tipo de metodología, son tres: especular, colaborar y aprender, estas fases o etapas corresponden al ciclo de vida, y se ejecutan para la construcción de un sistema o software, como se observa en la figura.

Figura 27

Esquema de la metodología Adaptive Software Development



Nota. En la figura "Esquema de la metodología Adaptive Software Development", observamos las etapas o fases correspondientes a este tipo de metodologías. Fuente: https://docplayer.es/54994652-Facultad-de-posgrados.html

ASD se compone de estas tres fases principales:

- *Especificación:* En esta fase se identifican los requisitos del sistema y se definen las funcionalidades que éste debe cumplir.

- *Ciclos de construcción:* Esta fase se enfoca en la construcción del sistema de manera incremental, dividiendo el proyecto en ciclos que se enfocan en las funcionalidades más importantes y críticas.
- *Despliegue:* En esta fase se lleva a cabo el despliegue del sistema, su instalación y su puesta en marcha.

ASD se enfoca en la adaptación continua del proceso de desarrollo, por lo que se requiere de una comunicación efectiva y constante con los usuarios y con el equipo de desarrollo. Además, se debe contar con una planificación flexible que permita adaptarse a los cambios en los requisitos y a las necesidades del negocio.

La metodología Adaptive Software Development es una metodología ágil que se enfoca en la adaptación continua del proceso de desarrollo de software. En el análisis de sistemas, ASD se enfoca en la recopilación de requisitos, en el diseño y en la planificación del sistema de manera iterativa e incremental, con el fin de adaptarse a los cambios y necesidades del negocio.

3.2 PROCESO DEL SOFTWARE

El proceso del software es el conjunto de actividades y pasos necesarios para desarrollar software de manera sistemática y eficiente. El proceso del software es esencial para garantizar la calidad, la eficiencia y la satisfacción del usuario en el desarrollo de software. A menudo, el proceso del software sigue un ciclo de vida que consta de las siguientes fases:

- *Análisis de requisitos:* esta fase implica la identificación y definición de los requisitos del usuario, los objetivos del software y los requisitos del sistema.
- *Diseño:* en esta fase se crea el diseño del software, incluyendo la arquitectura, los componentes y la interfaz de usuario.
- *Implementación:* en esta fase se escribe el código del software, se realiza la integración de los componentes y se realiza la prueba del software.
- *Pruebas:* en esta fase se realiza la verificación y validación del software, asegurándose de que cumple con los requisitos y de que funciona correctamente.
- *Mantenimiento:* esta fase implica la corrección de errores, la mejora del rendimiento y la adición de nuevas funcionalidades al software existente.

Un proceso de software efectivo habilita a la organización a incrementar su productividad al desarrollar software:

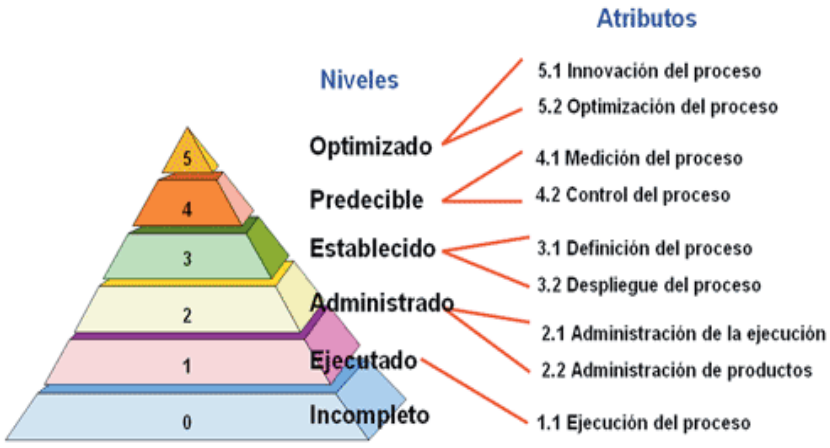
- Permite estandarizar esfuerzos, promover reúso, repetición y consistencia entre proyectos.
- Provee la oportunidad de introducir mejores prácticas de la industria.
- Permite entender que las herramientas deben ser utilizadas para soportar un proceso.
- Establece la base para una mayor consistencia y mejoras futuras.

Un proceso de software mejora los esfuerzos de mantenimiento y soporte:

- Define cómo manejar los cambios y liberaciones a sistemas de software existentes.
- Define cómo lograr la transición del software a la operación, y cómo ejecutar los esfuerzos de operación y soporte.

Figura 28

Niveles de capacidad del proceso del software



Nota. En la figura "Niveles de capacidad del proceso del software", observamos los diferentes niveles en los que puede encontrarse las diferentes etapas del proyecto. Fuente: https://sg.com.mx/revista/1/procesos-software

Actividades para un proyecto de software

La siguiente es una secuencia común de las actividades para un proyecto software:

1. Es necesario comprender la naturaleza del proyecto. Esto parece obvio, pero casi siempre lleva tiempo entender lo que desean los clientes.
2. Los proyectos requieren documentación desde el principio; es muy probable que esta documentación sufra muchos cambios. Por esta razón, desde el principio debe disponerse de una estrategia para mantener los documentos que se generen. Este proceso es denominado “Gestión de la Configuración”.
3. Hay que reunir los requisitos que ha de cumplir la aplicación. Gran parte de esta actividad es conversar con los “involucrados”.
4. Hay que analizar el problema, diseñar la solución y codificar los programas.
5. El producto inicial y final debe probarse en forma exhaustiva en todos sus aspectos.
6. Una vez entregado el producto, entra el modo “mantenimiento”, que incluye reparaciones y mejoras. Es una actividad que consume muchos recursos, a veces hasta el 65% de los recursos utilizados en el desarrollo de la aplicación.

El proceso del software puede variar según el enfoque y la metodología utilizados en el desarrollo de software. Algunas metodologías comunes incluyen el modelo en cascada, el modelo en espiral, la metodología ágil y la metodología DevOps.

3.2.1. FLUJOS DE PROCESOS

El flujo de procesos es una herramienta fundamental en el análisis de sistemas que permite representar de manera gráfica los pasos involucrados en un proceso para mejorar su eficiencia y efectividad. También conocido como diagrama de flujo de procesos o PFD, su objetivo principal es identificar los componentes del proceso, los flujos de información y los eventos que pueden ocurrir.

A través de la representación de los procesos con rectángulos, flujos de información con flechas, eventos con círculos y decisiones con rombos, el flujo de procesos facilita la identificación de cuellos de botella, puntos de falla y pasos redundantes en un proceso. Además, se pueden incluir notas y comentarios para agregar detalles adicionales sobre el proceso.

Una vez creado el flujo de procesos, se puede utilizar para analizar y mejorar el proceso. Por ejemplo, la eliminación de pasos redundantes o la realización de ajustes para evitar cuellos de botella puede aumentar la eficiencia del proceso. En definitiva, el flujo de

procesos es una herramienta útil para optimizar y mejorar la eficiencia de los procesos en un sistema.

Describe la manera en que están organizadas las actividades estructurales y las acciones y tareas que ocurren dentro de cada una con respecto de la secuencia y el tiempo. A continuación, se definen 4 tipos de procesos de software:

- Flujo de proceso lineal.
- Flujo de proceso iterativo.
- Flujo de proceso evolutivo.
- Flujo de proceso paralelo

Flujo de proceso lineal. - Es un tipo de diagrama de flujo utilizado en el análisis de sistemas que muestra la secuencia de pasos que deben seguirse en un proceso con un orden lineal. Este tipo de diagrama de flujo es comúnmente utilizado en la ingeniería de procesos y en la gestión de proyectos.

El flujo de proceso lineal se compone de una serie de bloques que representan las diferentes etapas del proceso. Cada bloque está conectado por una flecha que indica la

dirección del flujo de trabajo. La secuencia de bloques representa la secuencia en que se deben realizar las tareas. Los bloques pueden representar actividades, operaciones, decisiones, entradas, salidas y otros elementos del proceso. Es especialmente útil cuando se desea entender el flujo de trabajo secuencial y los pasos necesarios para completar una tarea, puede también ser utilizado para identificar ineficiencias o cuellos de botella en el proceso, o para planificar un proyecto o una tarea compleja. Podemos observar en la figura 29 las actividades del proceso de flujo lineal y como se ejecutan.

Figura 29

Flujo de Proceso Lineal

Nota. En la figura "Flujo de Proceso Lineal", observamos como se ejecutan las diferentes actividades del proceso.
Fuente: *https://ingenieriaensoftwarenathalyalava.wordpress.com/2015/04/18/procesos-del-software-y-modelo-cascada/*

Sin embargo, una limitación del flujo de proceso lineal, es que no es adecuado para representar procesos complejos con múltiples ramas y decisiones. El flujo de proceso lineal es un tipo de diagrama de flujo utilizado en el análisis de sistemas para representar la secuencia de pasos necesarios para completar un proceso en orden lineal. Es una herramienta útil para entender y planificar procesos sencillos y secuenciales, pero no es adecuado para procesos complejos con múltiples decisiones y ramificaciones.

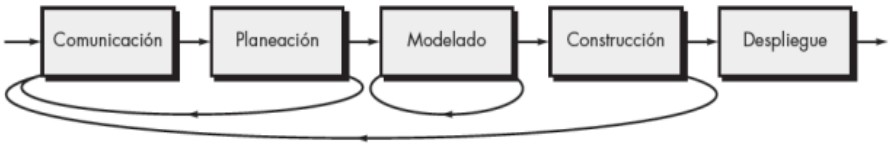
Flujo de proceso iterativo. - Es un tipo de diagrama de flujo utilizado en el análisis de sistemas que representa un proceso que se realiza de manera repetitiva, en ciclos o iteraciones. Este tipo de flujo de proceso se utiliza para modelar procesos en los que se deben realizar varias tareas o pasos en un orden específico, y en los que el proceso debe ser repetido varias veces para alcanzar un objetivo o un resultado final.

En un flujo de procesos iterativos, cada iteración se representa como una serie de pasos o bloques, que se realizan en un orden específico. El resultado de cada iteración se utiliza como entrada para la siguiente iteración, y así sucesivamente, hasta que se alcanza el objetivo final.

Los flujos de procesos iterativos son útiles en el análisis de sistemas porque permiten modelar procesos complejos que involucran múltiples ciclos de trabajo. También son útiles para representar procesos que se deben realizar en múltiples etapas y que requieren que ciertos pasos se repitan varias veces antes de avanzar al siguiente nivel.

Figura 30

Flujo de Proceso Iterativo



Nota. En la figura "Flujo de Proceso Iterativo", observamos como se ejecutan las diferentes actividades del proceso.
Fuente: https://ingenieriaensoftwarenathalyalava.wordpress.com/2015/04/18/procesos-del-software-y-modelo-cascada/

Un ejemplo común de un proceso iterativo es el proceso de desarrollo de software. En este caso, el flujo de proceso iterativo se utiliza para representar las diferentes etapas del proceso de desarrollo, como la planificación, el diseño, la codificación, las pruebas y la implementación. Cada iteración representa un ciclo de desarrollo completo, y el

proceso se repite varias veces hasta que se ha desarrollado un producto final completo y satisfactorio.

El flujo de procesos iterativos es un tipo de diagrama de flujo utilizado en el análisis de sistemas para representar procesos que se realizan de manera repetitiva en ciclos o iteraciones. Este tipo de flujo de proceso se utiliza para modelar procesos complejos que involucran múltiples ciclos de trabajo y es particularmente útil en el desarrollo de software y otros procesos de ingeniería.

Flujo de proceso evolutivo. - Es un enfoque de análisis de sistemas que se centra en la evolución gradual y el mejoramiento continuo del sistema a lo largo del tiempo. En este enfoque, el proceso de desarrollo del sistema se divide en varias fases o iteraciones, y se va mejorando y ajustando en cada una de ellas.

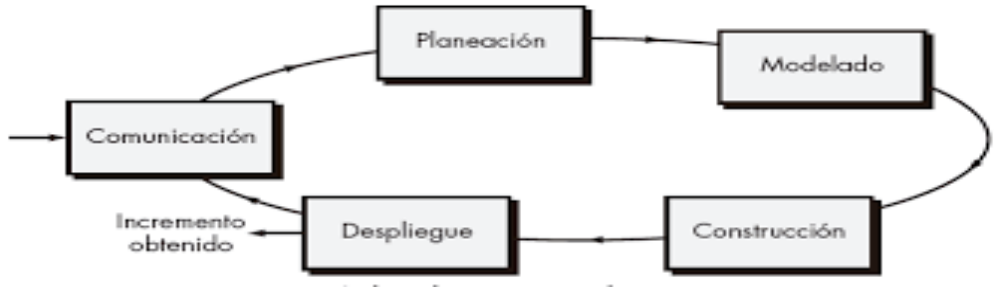
A diferencia del enfoque de desarrollo de sistemas lineal, en el que cada fase del proceso se completa antes de pasar a la siguiente, el flujo de proceso evolutivo implica la revisión y mejora continua del sistema en cada iteración. El proceso evolutivo se basa en la idea de que es difícil para los analistas de sistemas conocer todos los requisitos del sistema de antemano, y que el sistema evolucionará a medida que se descubran nuevas necesidades y se realicen cambios.

El flujo de proceso evolutivo comienza con un conjunto de requisitos iniciales, que se utilizan para desarrollar una versión inicial del sistema. Luego, se llevan a cabo pruebas y se recopilan comentarios para identificar los problemas y las áreas de mejora. Con base en estos comentarios, se realizan cambios y mejoras en el sistema, y se desarrolla una nueva versión.

Este proceso se repite varias veces, con cada iteración mejorando el sistema a partir de las lecciones aprendidas en la iteración anterior. El proceso evolutivo puede continuar durante varios ciclos, hasta que se logra un sistema satisfactorio y completo.

Figura 31

Flujo de Proceso Evolutivo



Nota. En la figura "Flujo de Proceso Evolutivo", observamos como se ejecutan las diferentes actividades del proceso.
Fuente:
https://www.uv.mx/personal/ermeneses/files/2018/02/Clase8_Modelos_de_procesos_de_desarrollo_de_softwareI.pdf

El enfoque del flujo de proceso evolutivo es especialmente útil en el desarrollo de sistemas complejos, en los que los requisitos no son completamente conocidos de antemano y se espera que evolucionen con el tiempo. Este enfoque permite a los analistas de sistemas adaptarse y ajustarse al sistema a medida que se desarrolla, lo que puede mejorar la calidad del sistema final y reducir el tiempo y los costos de desarrollo.

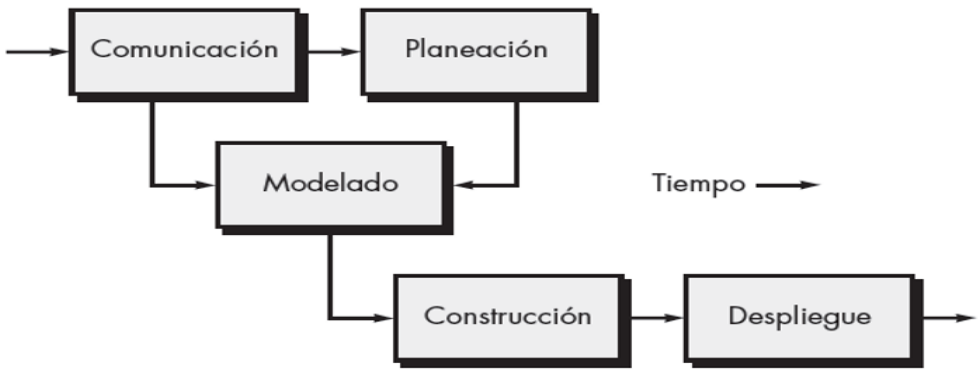
El flujo de proceso evolutivo es un enfoque de análisis de sistemas que se centra en la evolución gradual y el mejoramiento continuo del sistema a lo largo del tiempo. Este enfoque implica la revisión y mejora continua del sistema en cada iteración, y es especialmente útil en el desarrollo de sistemas complejos que tienen requisitos en constante evolución.

Flujo de proceso paralelo. – se refiere a un tipo de proceso que permite que varias tareas o actividades se realicen simultáneamente, en lugar de llevarse a cabo una tras otra de forma secuencial.

En un flujo de proceso paralelo, se dividen las actividades en diferentes subprocesos que se pueden realizar al mismo tiempo. Cada subproceso se asigna a un recurso o grupo de recursos, como un equipo o un departamento, y se ejecuta simultáneamente con los otros subprocesos. Una vez que se completan todos los subprocesos, se unen para formar el proceso completo, como observamos en la figura 32.

Figura 32

Flujo de Proceso Paralelo



Nota. En la figura "Flujo de Proceso Paralelo", observamos como se ejecutan las diferentes actividades del proceso.

Fuente:
https://www.uv.mx/personal/ermeneses/files/2018/02/Clase8Modelos_de_procesos_de_desarrollo_de_softwareI.pdf

Este enfoque puede ser especialmente útil en situaciones en las que el tiempo es un factor crítico, ya que permite que varias actividades se realicen simultáneamente, lo que puede acelerar el proceso completo. Sin embargo, también puede ser más complejo y difícil de gestionar que un flujo de proceso secuencial. Por lo tanto, es importante analizar cuidadosamente las necesidades y requisitos específicos de cada proyecto antes de decidir si utilizar o no un flujo de proceso paralelo en el análisis de sistemas.

RESUMEN DEL CAPÍTULO 3

El capítulo de "Modelos de Procesos y Metodologías" se enfoca en explicar los diferentes modelos y metodologías utilizados en la gestión de proyectos de tecnología de información (TI).

Se comienza describiendo los modelos de procesos de software más comunes, incluyendo el modelo en cascada, el modelo en espiral, el modelo V y el modelo ágil. Cada uno de estos modelos tiene su propia estructura y enfoque en la gestión del ciclo de vida del desarrollo de software.

Luego, se explican las diferentes metodologías utilizadas en la gestión de proyectos de TI, como la metodología de desarrollo de sistemas estructurados (SDLC), la metodología de gestión de proyectos de la Guía PMBOK, la metodología de desarrollo rápido de aplicaciones (RAD), entre otras. Cada una de estas metodologías tiene su propio enfoque y conjunto de prácticas recomendadas para la gestión de proyectos de TI.

Finalmente, se discute la importancia de elegir la metodología y modelo de proceso adecuados para cada proyecto de TI. Se enfatiza que la elección adecuada puede mejorar significativamente la eficiencia y efectividad del proyecto, y se debe considerar cuidadosamente los requisitos del proyecto, el equipo de proyecto y otros factores relevantes antes de seleccionar una metodología o modelo de proceso.

CAPÍTULO 4

TECNICAS DE RECOPILACIÓN DE DATOS Y ANÁLISIS DE DOCUMENTOS

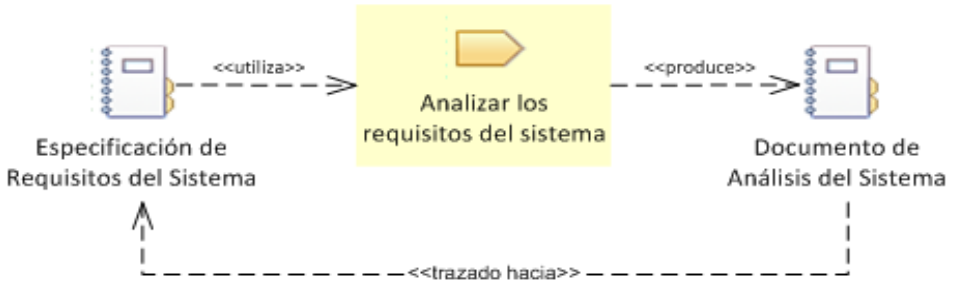
4.1 ANÁLISIS DE DOCUMENTOS

El análisis de documentos es una técnica utilizada dentro del análisis de sistemas de información, que consiste en examinar los documentos existentes en una organización, tales como informes, facturas, formularios, manuales de procedimientos, políticas y otros registros, con el fin de identificar información relevante para el diseño y desarrollo de sistemas de información.

El análisis de documentos es una parte importante del proceso de recopilación de requisitos en el desarrollo de sistemas de información, ya que permite a los analistas obtener una comprensión detallada de los procesos y procedimientos de una organización, así como de los requisitos de información necesarios para apoyarlos.

Figura 33

Análisis de Documentos



Nota. En la figura "Análisis de documentos", observamos la relación del documento de Análisis del Sistema con la Especificación de Requisitos del Sistema. Fuente: https://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/411

Durante el análisis de documentos, los analistas pueden identificar patrones y tendencias en la información, identificar áreas donde se necesitan mejoras o cambios, y determinar las necesidades de información de diferentes grupos de usuarios. También pueden identificar posibles problemas de calidad de datos, como datos incompletos o duplicados, y diseñar soluciones para abordar estos problemas.

El análisis de documentos es una técnica importante que ayuda a los analistas a comprender los requisitos de información de una organización y a diseñar sistemas de información que satisfagan esas necesidades.

4.1.2 DOCUMENTOS CUALITATIVOS

En el análisis de sistemas de información, los documentos cualitativos se refieren a los datos y documentos no numéricos que se utilizan para comprender y analizar un sistema de información. Estos documentos incluyen informes de usuarios, descripciones narrativas, entrevistas, notas de campo, transcripciones de reuniones y cualquier otro material que no se puede medir o cuantificar fácilmente.

Los documentos cualitativos son importantes en el análisis de sistemas de información porque ayudan a los analistas a comprender los procesos y procedimientos de un sistema de información desde la perspectiva de los usuarios y otros participantes. En diferentes fuentes que se realizan para determinar el enfoque de los documentos cualitativos podemos citar lo que mencionan diferentes autores:

Trabajar con investigación cualitativa en sistemas de información requiere saber identificar los datos a partir de los cuales surgen los diferentes indicadores, como pueden ser los sociales, que se encuentran en todas las áreas relacionadas con los sistemas de información. Donde también se incluyen habilidades y conocimientos de herramientas de investigación que deben ser consideradas y adoptadas para resolver diversos problemas sociales o humanos que se presenten. Estos instrumentos, aunque no han sido tratados en este trabajo, son diversos y variados, para poder enfrentar los diferentes tipos de problemas que pueden surgir.

(https://upcommons.upc.edu/bitstream/handle/2117/84898/R99-41.pdf, 1999, pág. 33)

Sobre el manejo de datos:

Estos datos pueden ser homogéneos o heterogéneos, por lo que debe definir claramente el método que se utilizará al trabajar con este tipo de datos. Esto es importante porque uno de los valores que aporta este estudio, es que permite a las empresas obtener información detallada sobre otros factores como las necesidades de los clientes, opiniones, ideas y planes. Evidentemente, esto les puede dar un valor añadido ya que les

permite anticiparse a las tendencias futuras en diferentes mercados como el del automóvil, la moda o el retáis. (Cegid Ekon, 2020)

Algunas técnicas comunes para analizar documentos cualitativos en el contexto de un sistema de información incluyen la codificación de datos, la categorización de temas y la identificación de patrones y tendencias a través del análisis de contenido. Estos métodos pueden ayudar a los analistas a identificar problemas y oportunidades de mejora en un sistema de información, así como a desarrollar soluciones y recomendaciones basadas en datos concretos. Existen varias técnicas para la obtención de datos cualitativos en análisis de sistemas, a continuación, algunas:

Figura 34

Como recolectar datos cualitativos

Nota. En la figura "Cómo recolectar datos cualitativos", como podemos ver las diferentes técnicas para la recolección de datos cualitativos. Fuente: https://reisdigital.es/datos-e-informacion/datos-cualitativos-ordinales/

Entrevistas: Las entrevistas son una técnica común para obtener datos cualitativos. En esta técnica, se realiza una serie de preguntas a los individuos o grupos de individuos que están involucrados en el sistema que se está analizando. Las preguntas pueden ser abiertas o cerradas, y se utilizan para obtener información sobre sus percepciones, opiniones, experiencias, actitudes y comportamientos relacionados con el sistema.

Observación: La observación es una técnica en la que el analista de sistemas observa y registra el comportamiento de las personas involucradas en el sistema en su entorno natural. Esta técnica se utiliza para obtener información sobre cómo se comportan las personas en situaciones específicas y cómo interactúan con el sistema.

Grupos focales (focus group): Los grupos focales son una técnica en la que se reúne a un grupo de personas que están involucradas en el sistema y se les hace preguntas abiertas

para obtener información sobre sus opiniones y experiencias relacionadas con el sistema. Esta técnica se utiliza para obtener una perspectiva grupal y entender cómo las personas interactúan entre sí en el contexto del sistema.

Análisis de documentos: El análisis de documentos es una técnica que implica revisar y analizar documentos relacionados con el sistema, como manuales, políticas, procedimientos y otros documentos relevantes. Esta técnica se utiliza para obtener información sobre cómo se estructura el sistema y cómo se llevan a cabo las actividades relacionadas con el sistema.

Análisis de casos: El análisis de casos es una técnica en la que se estudian casos específicos relacionados con el sistema para comprender cómo se han manejado y resuelto situaciones similares en el pasado. Esta técnica se utiliza para obtener información sobre cómo el sistema ha funcionado en situaciones específicas y cómo se pueden mejorar las soluciones en el futuro.

Técnica Delphi: es una técnica de investigación utilizada para recopilar y analizar opiniones de expertos en un tema específico. Es un proceso iterativo y anónimo que se utiliza para llegar a un consenso sobre un tema en particular. En esta técnica se selecciona un grupo de expertos en el tema y se les envía una serie de cuestionarios o encuestas. Los expertos completan los cuestionarios y envían sus respuestas de forma anónima al coordinador del proceso. El coordinador analiza las respuestas y prepara un resumen de los resultados. Luego, se envía una segunda ronda de cuestionarios a los expertos, en la que se les informa de los resultados de la primera ronda y se les pide que revisen sus respuestas en consecuencia. Este proceso continúa hasta que se llega a un consenso o hasta que se alcanza un límite predeterminado de rondas.

Es una técnica utilizada en una amplia gama de campos, como la planificación estratégica, la toma de decisiones, la predicción y la evaluación de riesgos. Se utiliza cuando no existe un consenso claro sobre un tema y se requiere la opinión de expertos para llegar a una conclusión. La técnica Delphi es útil porque permite que los expertos den su opinión de forma anónima y evita que cualquier experto influya en las respuestas de otros. Además, el proceso iterativo ayuda a reducir la variabilidad de las respuestas y aumenta la precisión de los resultados.

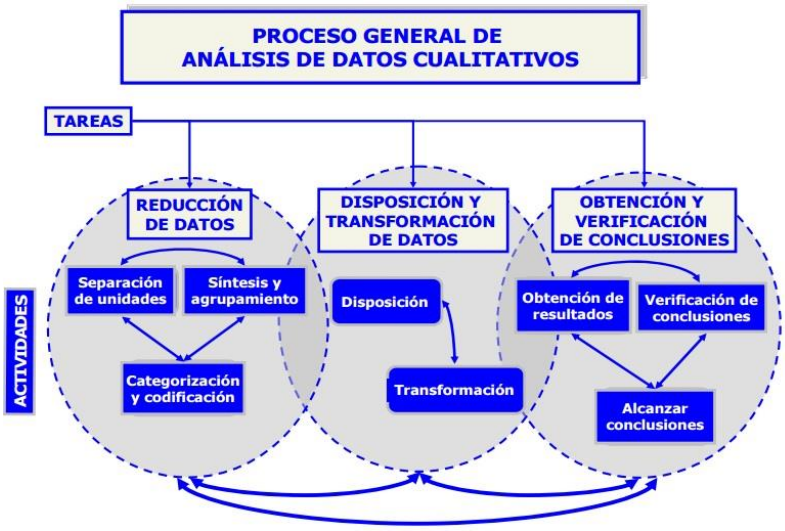
Cada técnica tiene sus propias ventajas y desventajas, y es importante seleccionar la técnica adecuada según los objetivos del análisis y el contexto del sistema que se está analizando.

Fases del análisis cualitativo

El análisis de datos cualitativos enmarcado en el análisis de sistemas, se divide en varias fases, que pueden variar dependiendo del enfoque o la metodología utilizada. Sin embargo, a continuación, se presenta una posible estructura de las fases del análisis de datos cualitativos utilizada para el análisis de sistemas;

Figura 35

Fases del análisis cualitativo



Nota. En la figura "Fases del Análisis cualitativo", observamos las diferentes fases del Análisis cualitativo, tareas y actividades. Fuente: http://el-diariodecampo.blogspot.com/2016/05/analisis-de-datos.html

Preparación de los datos: En esta fase, se realiza una primera revisión y organización de los datos recolectados. Los datos se codifican y se clasifican según temas, categorías o patrones que surgen del material recopilado. También se realiza una primera revisión de la calidad de los datos y se hace una evaluación de la consistencia entre las diferentes fuentes de información.

Desarrollo de categorías: Se lleva a cabo una revisión más detallada y se desarrollan categorías o temas emergentes a partir de los datos. Las categorías se establecen a través del análisis comparativo de los datos y se definen a medida que se identifican los patrones recurrentes en el material recopilado. Se busca reducir los datos y agruparlos en categorías o temas para facilitar su análisis posterior.

Análisis de los datos: En esta fase, se analiza la información recopilada para identificar patrones, tendencias, relaciones y conexiones entre las diferentes categorías. Se lleva a

cabo una exploración más profunda de los datos y se elaboran hipótesis o interpretaciones preliminares a partir de los resultados del análisis. También se identifican las limitaciones y las lagunas en los datos para guiar la recolección de información adicional, si es necesario.

Validación de los resultados: En esta fase, se valida la interpretación de los resultados y se verifica la validez y la confiabilidad de los hallazgos. Esto se puede hacer a través de la triangulación de datos, la revisión por pares, el muestreo negativo, la auditoría del proceso de análisis y la reflexividad.

Presentación de los resultados: En esta fase, se elaboran informes que presentan los resultados del análisis y se comunican a los diferentes interesados en el sistema analizado. Estos informes pueden incluir gráficos, tablas, citas y otros elementos visuales que ayuden a ilustrar los hallazgos del análisis. Además, se pueden incluir recomendaciones o sugerencias para mejorar el sistema o para futuras investigaciones.

Cada fase es importante y está interrelacionada con las demás, por lo que un análisis de datos cualitativos efectivo requiere de un proceso riguroso y sistemático.

4.1.3 DOCUMENTOS CUANTITATIVOS

En el análisis de sistemas, los documentos cuantitativos son aquellos documentos que contienen información numérica y datos medibles. Estos documentos pueden incluir informes financieros, estadísticas, encuestas, cuestionarios, registros de transacciones y otros tipos de datos cuantificables. Se utilizan para proporcionar información objetiva y medible sobre el desempeño del sistema. La información contenida en estos documentos se puede utilizar para identificar patrones, tendencias y relaciones en los datos, mismos que pueden ayudar a los analistas a comprender mejor el sistema y a tomar decisiones informadas.

En el análisis de sistemas, los documentos cuantitativos se pueden utilizar en combinación con los documentos cualitativos, que contienen información más descriptiva y subjetiva. Juntos, estos documentos pueden proporcionar una comprensión más completa y profunda del sistema que se está analizando.

Es importante tener en cuenta que los documentos cuantitativos pueden tener limitaciones, como la posibilidad de que los datos no sean representativos o que los métodos de medición no sean precisos. Por lo tanto, es importante que los analistas de sistemas evalúen cuidadosamente la calidad y la fiabilidad de los documentos cuantitativos antes de utilizarlos en el análisis.

Tipos de documentos cuantitativos

Existen varios tipos de documentos cuantitativos que se utilizan en el análisis de sistemas. A continuación, se presentan algunos ejemplos:

Figura 36

Técnicas de recolección de datos cuantitativos

Nota. En la figura "Cómo recolectar datos cuantitativos", como podemos observar en la figura se detallan las diferentes técnicas para la recolección de datos cualitativos. Fuente: https://reisdigital.es/datos-e-informacion/datos-cualitativos-ordinales/

Informes financieros: Estos documentos proporcionan información sobre los ingresos, los gastos y los balances financieros de una organización o sistema. Pueden incluir estados de resultados, balances generales, estados de flujo de efectivo y otros informes financieros relevantes.

Estadísticas: Estos documentos contienen información numérica que se utiliza para medir y analizar diferentes aspectos del sistema. Por ejemplo, pueden incluir estadísticas sobre la producción, el consumo, las ventas, los precios, el empleo, la demografía y otros indicadores relevantes.

Encuestas y cuestionarios: Estos documentos se utilizan para recopilar información de los usuarios, los empleados, los clientes u otras partes interesadas en el sistema. Pueden incluir preguntas cerradas, como opciones múltiples o escalas de Likert, o preguntas abiertas que permiten respuestas libres.

Registros de transacciones: Estos documentos contienen información sobre las transacciones realizadas en el sistema. Por ejemplo, pueden incluir registros de ventas,

facturas, recibos, registros de inventario y otros documentos que proporcionan información detallada sobre las transacciones realizadas.

Informes de desempeño: Estos documentos proporcionan información sobre el desempeño del sistema en diferentes áreas, como la eficiencia, la calidad, la productividad y la rentabilidad. Pueden incluir informes de auditoría, informes de evaluación de proyectos y otros informes relevantes.

Es importante tener en cuenta que la selección de los documentos adecuados dependerá de los objetivos y las necesidades específicas del análisis.

Datos cualitativos vs. Datos cuantitativos

En análisis de sistemas, los datos cualitativos y cuantitativos pueden desempeñar roles importantes y complementarios. Tomando en cuenta que los datos cuantitativos son aquellos que se pueden medir o contar numéricamente. Por ejemplo, el número de usuarios de un sitio web, la cantidad de ventas de un producto, el tiempo que tarda un proceso en completarse, etc. Estos datos suelen ser fáciles de analizar y se pueden utilizar para realizar cálculos estadísticos y gráficos.

Por otro lado, los datos cualitativos se refieren a información no numérica que se recopila a través de observaciones, entrevistas, encuestas y otros métodos. Estos datos se centran en la calidad o naturaleza de algo, en lugar de en su cantidad. Por ejemplo, las opiniones de los usuarios sobre un producto, las descripciones de los procesos de negocio, las expectativas de los clientes, etc. Los datos cualitativos pueden proporcionar una visión más profunda de las percepciones y experiencias de los usuarios, así como de los factores subyacentes que influyen en su comportamiento.

Figura 37

Datos cualitativos vs datos cuantitativos

Nota. En la figura "Datos cualitativos vs. cuantitativos", como podemos observar se detalla de forma gráfica como se diferencian los tipos de datos. Fuente: https://reisdigital.es/ejemplos/datos-cuantitativos-ejemplos/

Ambos tipos de datos pueden ser valiosos en el análisis de sistemas. Los datos cuantitativos pueden ayudar a identificar patrones y tendencias, mientras que los datos cualitativos pueden proporcionar información más detallada sobre las percepciones y experiencias de los usuarios. En general, es recomendable utilizar una combinación de ambos tipos de datos para obtener una visión completa y detallada de los sistemas y procesos.

RESUMEN DEL CAPÍTULO 4

Las técnicas de recopilación de datos y análisis de documentos son fundamentales para la investigación y análisis de datos en diversas áreas. Entre las técnicas de recopilación de datos se encuentran la observación, las entrevistas, los cuestionarios, la revisión de documentos y la investigación de campo. Cada técnica tiene sus propias ventajas y desventajas y debe ser seleccionada de acuerdo con los objetivos de la investigación y las características de la muestra.

El análisis de documentos es una técnica que implica la recopilación y revisión de documentos relevantes, como informes, registros, notas y otros documentos escritos. Esta técnica es útil para recopilar datos históricos y actuales, y para proporcionar una El análisis de documentos cualitativo se enfoca en la interpretación de los documentos para descubrir patrones, temas y tendencias. Esta técnica utiliza métodos de análisis temático y de contenido para identificar y analizar los temas y conceptos principales. Por otro lado, el análisis de documentos cuantitativo implica la recopilación de datos numéricos de los documentos y su análisis estadístico. Esta técnica es útil para identificar patrones y tendencias numéricas en grandes conjuntos de datos.

En resumen, las técnicas de recopilación de datos y análisis de documentos son herramientas importantes para la investigación y análisis de datos. La selección adecuada de estas técnicas dependerá de los objetivos de la investigación y las características de la muestra. El análisis de documentos puede ser realizado de manera cualitativa o cuantitativa, y cada método proporciona diferentes enfoques y resultados para el análisis de datos. visión general del tema en estudio. El análisis de documentos puede ser realizado de manera cualitativa o cuantitativa, dependiendo de los objetivos de la investigación.

REFERENCIAS

Baltazar, F. G. (2008). *Blogpost*. Obtenido de http://ramcesdj1.blogspot.com/

Berzal, F. (20 de Marzo de 2023). Obtenido de https://elvex.ugr.es/idbis/db/docs/lifecycle.pdf

Baltazar, F. G. (2008). Blogpost. Obtenido de http://ramcesdj1.blogspot.com/

Berzal, F. (20 de Marzo de 2023). Obtenido de https://elvex.ugr.es/idbis/db/docs/lifecycle.pdf

Cegid Ekon. (3 de Julio de 2020). Ekon. Obtenido de https://www.ekon.es/blog/tipos-analisis-datos-cualitativos/

Dennis, A., Wixom, B. H., & Roth, R. M. (2015). Sistemas de información: análisis y diseño (5ª ed.). Wiley.

Eriksson. (6 de Febrero de 2017). pmoinformatica.com. Obtenido de http://www.pmoinformatica.com/2017/02/requerimientos-funcionales-ejemplos.html

https://upcommons.upc.edu/bitstream/handle/2117/84898/R99-41.pdf. (1999). Obtenido de https://upcommons.upc.edu/bitstream/handle/2117/84898/R99-41.pdf.

Kendall, K. E., & Kendall, J. E. (2011). Análisis y diseño de sistemas (8ª ed.). Pearson.

Milano, P. (2007). Seguridad en el ciclo de vida del desarrollo de software. Obtenido de www.cybsec.com/upload/cybsec_Tendencias2007_Seguridad_SDLC.pdf

neuvoo. (2017). Neuvoo. Obtenido de https://neuvoo.com.mx/neuvooPedia/es/desarrollador-de-software/

Omaña, N. (22 de Mayo de 2010). Blogger. Obtenido de http://nestor-omana-sistemasinformacion.blogspot.com/2010/05/tipos-de-sistemas-segun-el-nivel-de.html

Pressman, R. S. (2014). Ingeniería del software: un enfoque práctico (7ª ed.). McGraw-Hill.

Puentes, M. (4 de Mayo de 2015). Blogspot. Obtenido de http://estructuratelentohumano.blogspot.com/2015/05/requerimientos-de-los-sistemas-de.html

Requeridos Blog. (20 de Abril de 2018). Medium. Obtenido de https://medium.com/@requeridosblog/requerimientos-funcionales-y-no-funcionales-ejemplos-y-tips-aa31cb59b22a

Satzinger, J. W., Jackson, R. B., & Burd, S. D. (2011). Análisis y diseño de sistemas de información (5ª ed.). Cengage Learning.

Sedici. (28 de Marzo de 2023). Sedici. Obtenido de http://sedici.unlp.edu.ar/bitstream/handle/10915/4060/I_-_Sistemas_de_informaci%C3%B3n.pdf?sequence=5&isAllowed=y

Senn, J. A. (2010). Análisis y diseño de sistemas de información (4ª ed.). Cengage Learning.

Sommerville, I. (2005).Cegid Ekon. (3 de Julio de 2020). *Ekon*. Obtenido de https://www.ekon.es/blog/tipos-analisis-datos-cualitativos/

Eriksson. (6 de Febrero de 2017). *pmoinformatica.com*. Obtenido de http://www.pmoinformatica.com/2017/02/requerimientos-funcionales-ejemplos.html

https://upcommons.upc.edu/bitstream/handle/2117/84898/R99-41.pdf. (1999). Obtenido de https://upcommons.upc.edu/bitstream/handle/2117/84898/R99-41.pdf.

Milano, P. (2007). *Seguridad en el ciclo de vida del desarrollo de software*. Obtenido de www.cybsec.com/upload/cybsec_Tendencias2007_Seguridad_SDLC.pdf

Neuvoo. (2017). *Neuvoo*. Obtenido de https://neuvoo.com.mx/neuvooPedia/es/desarrollador-de-software/

Omaña, N. (22 de Mayo de 2010). *Blogger*. Obtenido de http://nestor-omana-sistemasinformacion.blogspot.com/2010/05/tipos-de-sistemas-segun-el-nivel-de.html

Puentes, M. (4 de Mayo de 2015). *Blogspot*. Obtenido de http://estructuratelentohumano.blogspot.com/2015/05/requerimientos-de-los-sistemas-de.html

Requeridos Blog. (20 de Abril de 2018). *Medium*. Obtenido de https://medium.com/@requeridosblog/requerimientos-funcionales-y-no-funcionales-ejemplos-y-tips-aa31cb59b22a

Sedici. (28 de Marzo de 2023). *Sedici*. Obtenido de http://sedici.unlp.edu.ar/bitstream/handle/10915/4060/I_-_Sistemas_de_informaci%C3%B3n.pdf?sequence=5&isAllowed=y

Sommerville, I. (2005). *Ingeniería del Software*. España: Pearson Educación. Obtenido de https://www.uv.mx/personal/fcastaneda/files/2015/08/F_Capitulo_5_Requerimientos_del_software.pdf

Valerdi, R. (2013). Análisis y diseño de sistemas de información. Pearson.

Vera, M. (2021). *Metodología gestión de requerimientos*. Obtenido de https://sites.google.com/site/metodologiareq/capitulo-ii/tecnicas-para-identificar-requisitos-funcionales-y-no-funcionales

Visure Solutions. (2023). *visure*. Obtenido de https://visuresolutions.com/es/blog/non-functional-requirements/

Whitten, J. L., Bentley, L. D., & Dittman, K. C. (2012). Sistemas de información: análisis y diseño (7ª ed.). McGraw-Hill.