Universidad privada Domingo Savio

Análisis de Sistemas

Estudiante: Laura Andrea Rivero Parada

Carrera: Ingeniería en sistema

Docente: Hugo Arnaldo Guzman Centellas

Santa Cruz – Bolivia

INDICE

Análisis de Sistema

1.1 Sistema de información

Un sistema de información es un conjunto de elementos que interactúan entre sí para recolectar, procesar, almacenar y distribuir información con el fin de apoyar la toma de decisiones y controlar los procesos de una organización. Estos sistemas combinan hardware, software, datos, personas y procedimientos para gestionar información. Los sistemas de información pueden incluir desde simples bases de datos hasta aplicaciones complejas que soportan todos los aspectos de una empresa, como la planificación de recursos empresariales (ERP) o la gestión de relaciones con clientes (CRM).

1.2 Personal Involucrado en el Desarrollo

El desarrollo de sistemas de información implica la colaboración de diferentes roles especializados:

Usuarios finales: Son las personas que utilizan el sistema en su vida diaria. Proporcionan retroalimentación esencial para mejorar el sistema.

Analistas de sistemas: Se encargan de estudiar las necesidades de la organización y proponer soluciones tecnológicas adecuadas.

Desarrolladores/Programadores: Diseñan y construyen el software o las aplicaciones que forman parte del sistema de información.

Gestores de proyectos: Son responsables de planificar, ejecutar y cerrar proyectos, asegurándose de que se cumplan los plazos, el presupuesto y los requisitos.

Administradores de bases de datos: Mantienen y gestionan los datos dentro del sistema, garantizando su integridad y seguridad.

Especialistas en seguridad: Encargados de proteger el sistema de posibles ataques, garantizar la privacidad de los datos y cumplir con las normativas de seguridad.

1.3. Procesos, metodologías y enfoques de desarrollo

Este apartado se refiere a los distintos procesos, metodologías y enfoques que se utilizan para desarrollar un sistema de información, asegurando que el sistema sea diseñado, construido, probado e implementado de manera eficaz y eficiente. Cada uno de estos términos tiene un significado particular dentro del ciclo de vida del desarrollo de software, y su correcta aplicación es clave para garantizar que el proyecto cumpla con los objetivos planteados en tiempo y forma.

Procesos de desarrollo de sistemas

Los procesos de desarrollo son las diferentes fases o etapas que se siguen en el ciclo de vida de un proyecto de software. Estas fases se centran en organizar el trabajo y asegurar que todos los aspectos del desarrollo sean cubiertos, desde la planificación inicial hasta el mantenimiento del sistema. A continuación, se describen las principales fases del ciclo de vida de desarrollo de sistemas (SDLC, por sus siglas en inglés):

Planificación: Se definen los objetivos del sistema y se determina la viabilidad del proyecto. Esta fase involucra la definición de recursos, plazos, presupuesto y riesgos.
Análisis de requisitos: Se identifican las necesidades del negocio y los requisitos del sistema, tanto funcionales como no funcionales. Esta fase es crucial para entender qué necesita el usuario final.
Diseño del sistema: Se crea una solución técnica que satisfaga los requisitos. Esto incluye el diseño de la arquitectura del sistema, las bases de datos, la interfaz de usuario, etc.

Desarrollo o implementación: Los programadores desarrollan el código del sistema basado en el diseño aprobado. Se escribe el software que ejecuta las funciones requeridas.

Pruebas: El sistema es probado para asegurar que funcione según lo esperado, identificando y corrigiendo errores antes de su implementación.
Implementación: El sistema se pone en funcionamiento en el entorno real. A veces se hace en fases para minimizar el impacto en las operaciones del negocio.
Mantenimiento: Una vez que el sistema está en producción, se realizan ajustes y actualizaciones para corregir problemas, mejorar el rendimiento o adaptarlo a nuevas necesidades.

Metodologías de desarrollo de sistemas

Las metodologías son enfoques estructurados que definen cómo llevar a cabo las diferentes fases del desarrollo de sistemas. Estas metodologías proporcionan un marco de trabajo, herramientas y técnicas para desarrollar software de manera más organizada. Algunas de las metodologías más utilizadas incluyen:

Cascada: Es una metodología secuencial y lineal donde cada fase del desarrollo debe completarse antes de pasar a la siguiente. Aunque es una de las metodologías más antiguas, su rigidez puede ser un problema si se necesita hacer cambios una vez que una fase ha concluido.

Metodologías ágiles: Se centran en el desarrollo iterativo e incremental, donde se entregan pequeñas partes funcionales del sistema en cortos periodos de tiempo, llamados "sprints". Las metodologías ágiles, como **Scrum** y **Kanban**, permiten una mayor flexibilidad y adaptación a los cambios en los requisitos. En lugar de seguir un camino lineal, el desarrollo se ajusta continuamente según las necesidades del cliente o los usuarios finales.

Desarrollo rápido de aplicaciones (RAD): Enfocada en acelerar el proceso de desarrollo mediante la creación de prototipos funcionales rápidamente, lo que permite a los usuarios validar el sistema antes de que esté completamente terminado.

Modelo V: Es una extensión del modelo en cascada, en el que cada fase de desarrollo tiene una fase de prueba correspondiente. Esto asegura que las pruebas se planifiquen desde el inicio, minimizando problemas en etapas posteriores.

Enfoques de desarrollo de sistemas

Los enfoques de desarrollo son estrategias que determinan cómo se estructuran y gestionan los equipos, las tareas y los recursos a lo largo del proyecto. Los enfoques pueden variar dependiendo del tamaño del proyecto, la complejidad del sistema y los requerimientos específicos de la empresa. Algunos enfoques populares son:

Desarrollo orientado a objetos: Se centra en el uso de objetos y clases para estructurar el sistema, facilitando la reutilización de código y la creación de sistemas más flexibles y mantenibles.

Desarrollo basado en prototipos: Implica la creación de versiones tempranas y funcionales del sistema para que los usuarios puedan interactuar con él y dar retroalimentación. Esto permite realizar ajustes antes de que se complete el desarrollo total.

Desarrollo incremental: Se basa en desarrollar el sistema en pequeños módulos o partes que se completan en fases, cada una de las cuales añade una funcionalidad adicional hasta que el sistema esté completamente operativo.

Desarrollo dirigido por modelos (MDD): Es un enfoque que utiliza modelos visuales para definir la estructura y comportamiento del sistema, lo que permite la generación automática de código a partir de esos modelos. Facilita el diseño y mejora la consistencia entre el diseño y el producto final.

2.1. Concepto de análisis

El análisis de sistemas es el proceso de descomponer un sistema existente o planificado en sus componentes más pequeños para comprender su funcionamiento y determinar cómo puede ser mejorado. Durante esta fase, se identifican problemas, necesidades y oportunidades dentro del sistema, permitiendo la planificación de una solución más efectiva. Este proceso también incluye la evaluación de los requisitos de los usuarios y la identificación de las especificaciones que el nuevo sistema debe cumplir.

Principales objetivos del análisis:

- Entender a fondo el problema o necesidad del negocio.
- Identificar los requisitos del sistema.
- Evaluar las alternativas tecnológicas disponibles.
- Definir el alcance del sistema.

2.2. Técnicas para el relevamiento de requisitos

El relevamiento de requisitos es el proceso mediante el cual se recopila y documenta la información necesaria para el desarrollo del sistema. A continuación, se describen algunas de las técnicas más comunes:

Entrevistas: Permiten obtener información directa de los usuarios o stakeholders. Pueden ser estructuradas, con preguntas definidas de antemano, o abiertas, dejando más libertad para que los entrevistados expliquen sus necesidades y problemas.

Cuestionarios: Consisten en formularios con preguntas claras y concisas distribuidos entre los usuarios para recopilar información de una manera rápida y eficiente. Se suelen utilizar cuando hay un gran número de usuarios.

Observación directa: Implica ver cómo los usuarios realizan sus tareas en el entorno real de trabajo. Esto ayuda a identificar flujos de trabajo ineficientes y áreas que necesitan mejora.

Prototipos: Crear una versión preliminar del sistema permite que los usuarios interactúen con él y proporcionen feedback temprano sobre las funcionalidades y el diseño.

Análisis de documentos: Examina los documentos actuales (manuales, reportes, formularios) para entender los flujos de información y procesos existentes.

2.3. Requerimientos funcionales y no funcionales

Requisitos Funcionales: Los requisitos funcionales son los que definen qué hará exactamente el sistema. Describen las acciones y funcionalidades que debe realizar para cumplir con su propósito.

Ejemplos:
Permitir registro de usuarios: El sistema debe tener la capacidad de permitir a los usuarios crear cuentas ingresando sus datos.
Procesar facturas: El sistema debe ser capaz de generar y gestionar facturas electrónicas.
Sistema de ventas: Debe ser capaz de gestionar compras en línea, con funciones como carrito de compras, pasarela de pagos, etc. Estos requisitos son los que guían el desarrollo

del sistema. Si no se definen adecuadamente, el sistema podría no cumplir con lo que los usuarios necesitan.

Requisitos No Funcionales: Los requisitos no funcionales, por otro lado, describen cómo debe comportarse el sistema en aspectos relacionados con su rendimiento, seguridad, usabilidad, etc. No se trata de las funciones que debe realizar, sino de las cualidades que debe poseer.

Ejemplos:
Tiempo de respuesta: El sistema debe responder en menos de 3 segundos a cualquier solicitud del usuario.
Escalabilidad: El sistema debe ser capaz de manejar 1000 transacciones simultáneas sin afectar el rendimiento.
Seguridad: Todos los datos del usuario deben estar encriptados para garantizar la confidencialidad. Aunque el sistema funcione correctamente, si no cumple con estos requisitos, puede que no sea aceptado por los usuarios o no funcione bien en un entorno de producción real. La rapidez, seguridad y facilidad de uso son esenciales para el éxito de cualquier sistema.

2.4. Modelado de sistemas

El modelado de sistemas es una técnica utilizada para representar de manera visual cómo interactúan los componentes del sistema y cómo fluyen los datos a través de él. El propósito del modelado es proporcionar una representación clara del sistema para los desarrolladores, analistas y otros stakeholders. Existen varias técnicas y herramientas para modelar sistemas, algunas de las más comunes son:

Diagramas de flujo de datos (DFD): Representan el flujo de datos entre las distintas partes de un sistema. Son útiles para visualizar cómo se mueven los datos a través de procesos y entidades dentro del sistema.

Diagramas de casos de uso: Se utilizan para mostrar las interacciones entre los usuarios (actores) y el sistema, identificando las funcionalidades clave que estos utilizan.

Diagramas de entidad-relación (ER): Modelan las bases de datos y las relaciones entre las distintas entidades dentro del sistema, ayudando a entender cómo los datos están organizados y conectados.

Diagramas de actividad: Representan el flujo de trabajo o las acciones que deben ejecutarse dentro de un sistema. Ayudan a visualizar el comportamiento dinámico del sistema.

El modelado de sistemas es esencial porque proporciona una visión clara y estructurada de lo que se espera del sistema, facilitando la comunicación entre el equipo técnico y los stakeholders.

2.5. Modelado estructural

El modelado estructural se refiere a la representación estática de un sistema, describiendo sus componentes y cómo están organizados y relacionados entre sí. Este tipo de modelado se enfoca en la estructura del sistema, incluyendo sus objetos, clases y relaciones. Las técnicas más comunes de modelado estructural incluyen:

Diagramas de clases: Son utilizados en la metodología orientada a objetos para mostrar la estructura del sistema, describiendo las clases, sus atributos, métodos y las relaciones entre ellas.

Diagramas de componentes: Describen los diferentes módulos o componentes del sistema y cómo se relacionan entre sí. Esto es especialmente útil para sistemas complejos con múltiples subsistemas.

Diagramas de paquetes: Representan la agrupación lógica de clases o componentes, mostrando cómo están organizados dentro del sistema.

Este tipo de modelado es útil para entender la organización interna del sistema y asegurar que los elementos estructurales están diseñados de manera eficiente.

2.6. Modelado de comportamiento

El modelado de comportamiento describe cómo el sistema responde a las acciones y eventos externos. Representa la dinámica de interacción entre los componentes del sistema.

Diagramas de secuencia: Representan el intercambio de mensajes entre diferentes partes del sistema a lo largo del tiempo, lo que ayuda a entender la interacción entre componentes y el orden en que ocurren los eventos.

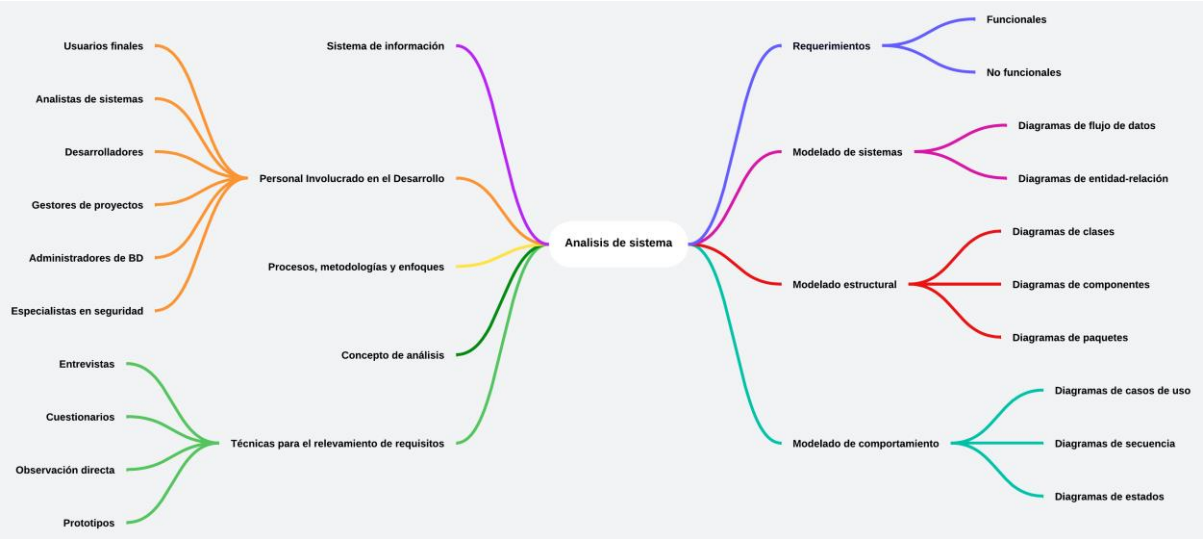
Diagramas de estados: Muestran los diferentes estados que puede tener un objeto dentro del sistema y los eventos que provocan las transiciones entre esos estados.

Bibliografía

- Guru99. (n.d.). *Requisitos funcionales versus no funcionales*. Recuperado de https://www.guru99.com/functional-vs-non-functional-requirements.html

- Corporación Universitaria Iberoamericana. (s.f.). *Modelado de sistemas complejos*. https://www.ibero.edu.co/blog/articulos/modelado-de-sistemas-complejos

Mapa Mental


Analisis de sistema
Sistema de información
Personal Involucrado en el Desarrollo
Procesos, metodologías y enfoques
Concepto de análisis
Técnicas para el relevamiento de requisitos
Usuarios finales
Analistas de sistemas
Desarrolladores
Gestores de proyectos
Administradores de BD
Especialistas en seguridad
Entrevistas
Cuestionarios
Observación directa
Prototipos
Requerimientos
Funcionales
No funcionales
Modelado de sistemas
Diagramas de flujo de datos
Diagramas de entidad-relación
Modelado estructural
Diagramas de clases
Diagramas de componentes
Diagramas de paquetes
Modelado de comportamiento
Diagramas de casos de uso
Diagramas de secuencia
Diagramas de estados