- # Conceptos de Sistema

1- ¿Qué es un sistema?

Se entiende por un sistema a un conjunto ordenado de componentes relacionados entre sí, ya se trate de elementos materiales o conceptuales, dotado de una estructura, una composición y un entorno particulares. Se trata de un término que aplica a diversas áreas del saber, como la física, la biología y la informática o computación.

2- ¿Como se clasifican los sistemas?

Según a la forma en que se constituyen pueden ser:

Sistemas físicos o concretos: compuestos cosas reales como equipos, maquinaria, objetos. Ejemplo: El hardware.
Sistemas abstractos: compuestos por conceptos, planes, hipótesis e ideas. Muchas veces solo existen en el pensamiento de las personas. Ejemplo: el software.

De acuerdo a su relación con el medio ambiente esto puede ser:

Abiertos: Sistemas que intercambian materia, energía o información con el ambiente.
Cerrados: Sistemas que no intercambian materia, energía o información con el ambiente.

Dependiendo de su naturaleza puede ser:
Concretos: Sistema físico o tangible.
Abstractos: Sistemas simbólicos o conceptuales.

Respecto a su origen estos pueden ser:
Naturales: Sistemas generados por la naturaleza.
Artificiales: Sistemas que son productos de la actividad humana, son concebidos y construidos por el hombre.

De acuerdo a sus relaciones pueden ser;
Simples: Sistemas con pocos elementos.
Complejos: Sistemas con numerosos elementos y relaciones.

Esta clasificación es respectiva porque depende del número de elementos y relación considerados. En la práctica y con base en límites psicológicos de la percepción y comprensión humanas, un sistema con más o menos siete elementos y relaciones se puede considerar simple.

Conforme su cambio de Tiempo:
Estáticos: Sistema que no cambia en el tiempo.
Dinámicos: Sistema que cambia en el tiempo.

Esta clasificación es relativa porque depende del periodo de tiempo definido para el análisis del Sistema.

Dependiendo al tipo de variable que lo definen;
Discretos: Sistema definido por variables discretas. Sistema definido por variables continúas.

3- ¿Qué es un sistema de información y como se clasifican?

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

En informática, los sistemas de información ayudan a administrar, recolectar, recuperar, procesar, almacenar y distribuir información relevante para los procesos fundamentales y las particularidades de cada organización.

Los sistemas de información, de manera general se pueden clasificar de tres formas según sus propósitos generales, en este sentido Peralta (2008) clasifica los sistemas de información en tres tipos fundamentales: (1) Sistemas transaccionales; (2) Sistemas de Soporte a la Toma de Decisiones, Sistemas para la Toma de Decisión de Grupo, Sistemas Expertos de Soporte a la Toma de Decisiones y Sistema de Información para Ejecutivos y (3) Sistemas estratégicos.

• Organizaciones

1- ¿Qué es una organización o una empresa?

es una unidad económico-social, integrada por elementos humanos, materiales y técnicos, que tiene el objetivo de obtener utilidades a través de su participación en el mercado de bienes y servicios. Para esto, hace uso de los factores productivos (trabajo, tierra y capital).

2- ¿Cómo se clasifican?

Principalmente, podemos clasificar a las organizaciones en dos grandes grupos, de acuerdo a su carácter y busca de rentabilidad:

Organización lucrativa: recibe su nombre ya que la misma posee fines de lucro, es decir que además de buscar una rentabilidad social, busca una rentabilidad económica. Este concepto lo podemos simplificar diciendo que son aquellas organizaciones que buscan un beneficio económico.
Organización no lucrativa: Son organizaciones sin fines de lucro, es decir que su objetivo principal no es la búsqueda de un beneficio económico.

Podemos clasificar la empresa desde distintos aspectos:
Según la titularidad del capital de la empresa:
Empresa privada: Es aquella que corresponde a particulares.
Empresa pública: El estado, u otros entes públicos son los propietarios.
Empresa mixta: La propiedad es compartida entre los particulares y el estado o entes públicos.

Según el número de propietarios:
Empresa unipersonal: La propiedad corresponde a una sola persona.
Empresa societaria: Los propietarios son dos o más personas que se asocian para desarrollar una actividad en común.

Según los sectores de actividad:
Empresa del sector primario: Su actividad se relaciona con los recursos naturales.
Empresas del sector secundario: Su actividad se relaciona con las industrias, o la transformación de bienes.
Empresas del sector terciario: Su actividad se relaciona con la prestación de servicios.

Según la dimensión de la empresa:
Grandes: están conformadas por más de 400 trabajadores.
Medianas: están conformadas por entre 50 y 400 trabajadores.
Pequeñas: poseen menos de 50 trabajadores.

- ## ¿Qué es un analista de sistema? ¿cuál es su papel en la empresa y cuáles son los atributos que debe reunir para desempeñar su roll?

El analista de sistemas o de tecnologías de la información es un profesional especializado del área de la ingeniería de software e informática, encargado del desarrollo de aplicaciones en lo que respecta a su diseño y obtención de los algoritmos, así como de analizar las posibles utilidades y modificaciones necesarias de los sistemas operativos para una mayor eficacia de un sistema informático. Otra misión de estas personas es dar apoyo técnico a los usuarios.

El Papel de este en una empresa es crear soluciones informáticas para problemas que la empresa en si no pueda resolver, también supervisar el optimo funcionamiento de los sistemas que la empresa posee, apoyar a los usuarios en el uso correcto del sistema, planifica y ejecuta la instalación de un nuevo sistema y da seguimiento a este buscando oportunidades de mejorarlo. Los atributos que debe tener para desempeñar este roll es actualizar tus conocimientos de forma constantes, ser proactivo y no tengas miedo a trabajar en equipo.

- **Cite y defina cuales son las etapas que hay que agotar para desarrollar un sistema de información (o Software).**

Análisis de requisitos

Extraer los requisitos de un producto de software es la primera etapa para crearlo. Mientras que los clientes piensan que ellos saben lo que el software tiene que hacer, se requiere de habilidad y experiencia en la ingeniería de software para reconocer requisitos incompletos, ambiguos o contradictorios. El resultado del análisis de requisitos con el cliente se plasma en el documento ERS, *Especificación de Requerimientos del Sistema*, cuya estructura puede venir definida por varios estándares, tales como CMM-I. Asimismo, se define un diagrama de Entidad/Relación, en el que se plasman las principales entidades que participarán en el desarrollo del software. La captura, análisis y especificación de requisitos (incluso pruebas de ellos), es una parte crucial; de esta etapa depende en gran medida el logro de los objetivos finales. Se han ideado modelos y diversos procesos de trabajo para estos fines. Aunque aun no está formalizada, ya se habla de la Ingeniería de Requisitos. La IEEE Std. 830-1998 normaliza la creación de las Especificaciones de Requisitos Software (Software Requirements Specification).

Diseño y arquitectura

Se refiere a determinar cómo funcionará de forma general sin entrar en detalles. Consiste en incorporar consideraciones de la implementación tecnológica, como el hardware, la red, etc. Se definen los Casos de Uso para cubrir las funciones que realizará el sistema, y se transforman las entidades definidas en el análisis de requisitos en clases de diseño, obteniendo un modelo cercano a la programación orientada a objetos.

Programación

Reducir un diseño a código puede ser la parte más obvia del trabajo de ingeniería de software, pero no es necesariamente la porción más larga. La complejidad y la duración de esta etapa está íntimamente ligada al o a los lenguajes de programación utilizados.

Pruebas

Consiste en comprobar que el software realice correctamente las tareas indicadas en la especificación. Una técnica de prueba es probar por separado cada módulo del software, y luego probarlo de forma integral, para así llegar al objetivo. Se considera una buena práctica el que las pruebas sean efectuadas por alguien distinto al desarrollador que la programó, idealmente un área de pruebas; sin perjuicio de lo anterior el programador debe hacer sus propias pruebas. En general hay dos grandes formas de organizar un área de pruebas, la primera es que esté compuesta por personal inexperto y que desconozca el tema de pruebas, de esta forma se evalúa que la documentación entregada sea de calidad, que los procesos descritos son tan claros que cualquiera puede entenderlos y el software hace las cosas tal y como están descritas. El segundo enfoque es tener un área de pruebas conformada por programadores con experiencia, personas que saben sin mayores indicaciones en que condiciones puede fallar una aplicación y que pueden poner atención en detalles que personal inexperto no consideraría.

Documentación

Todo lo concerniente a la documentación del propio desarrollo del software y de la gestión del proyecto, pasando por modelaciones (UML), diagramas, pruebas, manuales de usuario, manuales técnicos, etc.; todo con el propósito de eventuales correcciones, usabilidad, mantenimiento futuro y ampliaciones al sistema.

Mantenimiento

Mantener y mejorar el software para enfrentar errores descubiertos y nuevos requisitos. Esto puede llevar más tiempo incluso que el desarrollo inicial del software. Alrededor de 2/3 de toda la ingeniería de software tiene que ver con dar mantenimiento. Una pequeña parte de este trabajo consiste en arreglar errores, o *bugs*. La mayor parte consiste en extender el sistema para hacer nuevas cosas. De manera similar, alrededor de 2/3 de toda la ingeniería civil, arquitectura y trabajo de construcción es dar mantenimiento.

- **Dentro de cada una de las etapas para desarrollar un sistema de información (o Software), citar y definir cada uno de los pasos que deben de ser agotados.**

Planificación

La importante tarea a la hora de crear un producto de software es obtener los requisitos o el análisis de los requisitos. Los clientes suelen tener una idea más bien abstracta del resultado final, pero no sobre las funciones que debería cumplir el software. Es el paso previo al inicio de cualquier proyecto de desarrollo y sin dudas el más importante. En este se definen los requerimientos y funcionalidades que debe tener el software, mediante el trabajo en conjunto entre los desarrolladores, el departamento de ventas, los estudios de mercado y, fundamentalmente, el contacto con el cliente.

Implementación, pruebas y documentación

La implementación es parte del proceso en el que los ingenieros de software programan el código para el proyecto de trabajo que está en relación de las demandas del software, en esta etapa se realizan las pruebas de caja blanca y caja negra.

Las pruebas de software son parte esencial del proceso de desarrollo del software. Esta parte del proceso tiene la función de detectar los errores de software lo antes posible.

La documentación del diseño interno del software con el objetivo de facilitar su mejora y su mantenimiento se realiza a lo largo del proyecto. Esto puede incluir la documentación de un API, tanto interior como exterior. Prácticamente es como una receta de cocina.

Despliegue y mantenimiento

El despliegue comienza cuando el código ha sido suficientemente probado, ha sido aprobado para su liberación y ha sido distribuido en el entorno de producción.

Entrenamiento y soporte para el software es de suma importancia y algo que muchos desarrolladores de software descuidan. Los usuarios, por naturaleza, se oponen al cambio porque conlleva una cierta inseguridad, es por ello que es fundamental instruir de forma adecuada a los futuros usuarios del software.

El mantenimiento o mejora de un software con problemas recientemente desplegado, puede requerir más tiempo que el desarrollo inicial del software. Es posible que haya que incorporar código que no se ajusta al diseño original con el objetivo de solucionar un problema o ampliar la funcionalidad para un cliente. Si los costes de mantenimiento son muy elevados puede que sea oportuno rediseñar el sistema para poder contener los costes de mantenimiento.

- # El paradigma orientado a objetos, UML y el proceso unificado.

Los lenguajes de programación proporcionan mecanismos para implementar una filosofía o paradigma de programación. Un paradigma es una forma de entender y representar la realidad: un conjunto de teorías, estándares y métodos que, juntos, representan un modo de organizar el pensamiento, es decir, un modo de ver el mundo. Cada nuevo paradigma responde a una necesidad real de nuevas formas de afrontar problemas. A menudo un nuevo paradigma es creado como respuesta a las deficiencias de paradigmas anteriores.

Un paradigma de programación es una forma de conceptualizar en qué consiste la ejecución de un programa y cómo deben de estructurarse y organizarse las tareas que se llevaran a cabo en esa ejecución

Paradigma Orientado a Objetos

El paradigma orientado a objetos (OO) define los programas en términos de comunidades de objetos. Los objetos con características comunes se agrupan en clases (un concepto similar al de tipo abstracto de dato (TAD)). Los objetos son entidades que combinan un estado (es decir, datos) y un comportamiento (esto es, procedimientos o métodos). Estos objetos se comunican entre ellos para realizar tareas. Es en este modo de ver un programa donde este paradigma difiere del paradigma imperativo o estructurado, en los que los datos y los métodos están separados y sin relación. El paradigma OO surge para solventar los problemas que planteaban otros paradigmas, como el imperativo, con el objeto de elaborar programas y módulos más fáciles de escribir, mantener y reutilizar. Entre los lenguajes que soportan el paradigma OO están Smalltalk, C++, Delphi (Object Pascal), Java y C#.

El **lenguaje unificado de modelado** (**UML**, por sus siglas en inglés, *Unified Modeling Language*) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el *Object Management Group* (OMG).

Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y compuestos reciclados.

Es importante remarcar que UML es un "lenguaje de modelado" para especificar o para describir métodos o procesos. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo.

Objetos y Clases

una clase es una plantilla para la creación de objetos de datos según un modelo predefinido. Las clases se utilizan para representar entidades o conceptos, como los sustantivos en el lenguaje.

un objeto es una unidad dentro de un programa de computadores que consta de un estado y de un comportamiento, que a su vez constan respectivamente de datos almacenados y de tareas realizables durante el tiempo de ejecución.

Herencia

Es la facilidad mediante la cual la clase D hereda en ella cada uno de los atributos y operaciones de C, como si esos atributos y operaciones hubiesen sido definidos por la misma D. Por lo tanto, puede usar los mismos métodos y variables publicas declaradas en C. Los componentes registrados como "privados" (private) también se heredan, pero como no pertenecen a la clase, se mantienen escondidos al programador y sólo pueden ser accedidos a través de otros métodos públicos. Esto es así para mantener hegemónico el ideal de OOP.

Generalización, Agregación y Asociación.

Generalización

Es una relación de herencia. Se puede decir que es un relación “es un tipo de” (IS-A). En nuestro ejemplo: “un Autobús es un tipo de Medio de transporte”. Es entre una clase hija y su clase madre. En la codificación podemos encontrar la palabra “extends” que hace referencia a esta relación. Además podemos encontrar palabras claves tales como “this” y “super” (Java) o "self" y “parent” (PHP). Para darnos cuenta que existe una relación de este tipo involucrada.

Agregación:

Es una relación que se derivó de la asociación, por ser igualmente estructural, es decir que contiene un atributo, que en todos los casos, será una colección, es decir un Array, Vector, Collections, etc, y además de ello la clase que contiene la colección debe tener un método que agregue los elementos a la colección. También se puede leer como que un medio de transporte tiene varios pasajeros.

Asociación:

Es generalmente, una relación estructural entre clases, es decir, que, en el ejemplo, existe un atributo de la clase medio de transportes, que es del tipo Conductor. La navegalidad nos muestra donde está ubicado el atributo. Es decir, cual es la clase que tiene contiene el atributo si ésta no lo mostrase. La multiplicidad en una Asociación dice bastante, ya que de eso dependerá si el atributo, es una colección o simplemente una variable de referencia a un objeto.

El Proceso Unificado.

Es una metodología de desarrollo de software que está basado en componentes e interfaces bien definidas, y junto con el Lenguaje Unificado de Modelado (UML), constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

- **Definición y función de cada uno de las siguientes etapas del Ciclo de vida del desarrollado un sistema de información o Software; además citar y definir los pasos que hay que agotar en cada una de estas etapas:**

Comunicación

Este es el primer paso donde l usuario inicia la petición de un producto software determinado. Contacta al proveedor de servicios e intenta negociar las condiciones. Presenta su solicitud al proveedor de servicios aportando la organización por escrito.

Recolección de solicitudes

A partir de este paso y en adelante el equipo de desarrollo software trabaja para tirar adelante el proyecto. El equipo se reúne con varios depositarios de dominio del problema, e intentan conseguir la máxima cantidad de información possible sobre lo que requieren. Los requisitos se contemplan y agrupan en requisitos del usuario, requisitos funcionales y requisitos del sistema. La recolección de todos los requisitos se lleva a cabo como se especifica a continación -

- Estudiando el software y el sistema actual o obsoleto,
- Entrevistando a usuarios y a desarrolladores de Software,
- Consultando la base de datos o
- Recogiendo respuestas a través de cuestionarios.

Estudio de viabilidad

Después de la recolección de requisitos, el equipo idea un plan para procesar el software. En esta fase, el equipo analiza si el software puede hacerse para cubrir todos los requisitos del usuario y si hay alguna posibilidad de que el software ya no sea necesario. Se investiga si el proyecto es viable a nivel financiero, práctico, y a nivel tecnológico para que la organización acepte la oferta. Hay varios algoritmos disponibles, los cuales ayudan a los desarrolladores a concluir si el proyecto software es factible o no.

Análisis del sistema

En este pas los desarrolladores trazan su plan e intentan crear el mejor y más conveniente modelo de software para el proyecto. El análisis del sistema inclye el entendimiento de las limitaciones del producto Software; el aprendizaje de los problemas relacionados con el sistema; los cambios que se requieren en sistemas ya existentes con antelación, identificando y dirigiendo el impacto del proyecto a la organización y al personal, etc. El equipo del proyecto analiza las posibilidades del proyecto y planifica la temporalización y los recursos correspondientes.

Diseño de Software

El siguiente paso es diseñar el producto software con la ayuda de toda la información recogida sobre requisitos y análisis. Los inputs (aportaciones) de los usuarios y los resultados de la recogida de información hecha en la fase anterior serán las aportaciones base de la fase actual. El output (o resultado) de esta etapa toma la forma de 2 diseños; El diseño lógico y el diseño físico. Los ingenieros crean meta-data (Metadatos), Diagramas dilógicos, diagramas de flujo de datos, y en algunos casos pseudocódigos.

Codificación

Esta fase también se puede denominar 'fase de programación'. La implementación del diseño de software empieza con el lenguaje de programación más conveniente, y desarrollando programas ejecutables y sin errores de manera eficiente.

Pruebas

Se estima que el 50% de todos los procesos de desarrollo de software deberían ser evaluados. Los errores pueden arruinar el software tanto a nivel crítico y hasta el punto de ser eliminado. Las pruebas de Software se hacen mientras se codifica y suelen hacerlo los desarrolladores y otros expertos evaluadores a varios niveles. Esto incluye evaluación de módulos, evaluación del programa, evaluación del producto, evaluación interna y finalmente evaluación con el consumidor final. Encontrar errores y su remedio a tiempo es la llave para conseguir un software fiable.

Integración

El Software puede necesitar estar integrado con las bibliotecas, Bases de datos o con otro u otros programas. Esta fase del SDLC se focaliza en la integración del software con las entidades del mundo exterior.

Implementación

Aquí se instala el software en máquinas de clientes. A veces, el software necesita instalar configuraciones para el consumidor final con posterioridad. El Software se evalúa por su adaptabilidad y su portabilidad, en cuanto a las cuestiones relacionadas con la integración y conceptos asociados, se resuelven durante la implementación.

Mantenimiento y Funcionamiento

Esta fase confirma el funcionamiento del software en términos de más eficiencia y menos errores. Si se requiere, los usuarios se forman, o se les presta documentación sobre como operar y como mantenerlo en funcionamiento. El software se mantiene de forma temprana actualizando el código en acorde a los cambios que tienen lugar en entornos del usuario o tecnológicos. Esta fase puede que tenga que encarar retos originados por virus ocultos o problemas no identificados del mundo real.

Disposición

Con el paso del tiempo, puede que el software falle en su ejecución. Puede que se vuelva totalmente obsoleto o que necesite actualizaciones. De ahí surge una necesidad urgente de eliminar una parte importante del sistema. Esta fase incluye archivar datos y componentes software requeridos, cierre del sistema, planificación de la actividad de disposición y terminación de sistema en el momento final del sistema.

1- **Investigación preliminar de un sistema**

En esta etapa el analista se involucra en la identificación de los problemas, de las oportunidades y de los objetivos, esta etapa es crítica, ya que nadie desea perder el tiempo resolviendo el problema equivocado.

2- **Investigación detallada o profunda del sistema.**

Con el fin de implantar un sistema de información comercial, es necesario llevar a cabo un estudio de factibilidad. El primer paso del estudio es hacer un análisis del sistema actual (es decir un Análisis de sistemas). Este primer paso culmina en
un informe de investigación exploratoria que es revisado por la alta gerencia para determinar si es factible o no el llevar a cabo el proyecto de sistema. Diseños de sistemas es el segundo paso y el más creativo en la realización del estudio de factibilidad, comprende la determinación de los requerimientos del nuevo sistema, esto incluye el trabajar con las personas y resolver los problemas de las áreas.
El tercer y último paso del estudio de factibilidad es la selección del equipo una ves que el equipo ha pasado a formar parte integral del nuevo sistema de información de la empresa, al a que se le conoce como implantación de sistemas, se le debe revisar periódicamente con el objeto de hacer mejoras.