Nociones de ingeniería de software

Molina Ríos Jimmy, Honores Tapia Joofre, Zea Ordoñez Mariuxi

Universidad Técnica de Machala

Nociones de Ingeniería de Software

Ing. César Quezada Abad, MBA
Rector

Ing. Amarilis Borja Herrera, Mg. Sc.
Vicerrectora académica

Soc. Ramiro Ordóñez Morejón, Mg. Sc.
Vicerrector administrativo

COORDINACIÓN EDITORIAL
VICERRECTORADO ACADÉMICO

Tomás Fontaines-Ruiz, PhD.
Investigador Becario Prometeo-UTMACH
Asesor Del Programa de Reingeniería

Ing. Karina Lozano Zambrano
Coordinadora editorial

Ing. Jorge Maza Córdova, Ms.
Ing. Cyndi Aguilar
Equipo de Publicaciones

Nociones de Ingeniería de Software

Molina Ríos Jimmy Rolando

Honores Tapia Joofre Antonio

Zea Ordóñez Mariuxi Paola

UNIVERSIDAD TÉCNICA DE MACHALA

2015

Dedicatoria

Dedicamos este libro primeramente a Dios, que fue nuestra inspiración y fortaleza espiritual, también a todos aquellos que nos apoyaron moralmente y todos quienes participaron durante toda la realización del presente libro.

Nuestra familia, quienes siempre estuvieron presentes de inicio a fin, que sabemos que representa el pilar fundamental de nuestras vidas, pues es a ellos a quienes se lo debemos por su apoyo incondicional para la realización del mismo.

Los Autores

Primera edición 2015

ISBN: 978-9942-24-007-1

D.R. © 2015, UNIVERSIDAD TÉCNICA DE MACHALA
Ediciones UTMACH
Km. 5 1/2 Vía Machala Pasaje
www.utmachala.edu.ec

Este texto ha sido sometido a un proceso de evaluación por pares externos con base en la normativa editorial de la UTMACH.

Portada:
Concepto editorial
Samanta Cabezas (est. comunicación social)
Fotografía: Dir. de Comunicación UTMACH
Diseño, montaje y producción editorial: UTMACH

Impreso y hecho en Ecuador
Printed and made in Ecuador

Advertencia: "Se prohíbe la reproducción, el registro o la transmisión parcial o total de esta obra por cualquier sistema de recuperación de información, sea mecánico, fotoquímico, electrónico, magnético, electroóptico, por fotocopia o cualquier otro, existente o por existir, sin el permiso previo por escrito del titular de los derechos correspondientes".

Prólogo

La innovación en el hardware de los últimos años ha sido notoria, y podría parecer que los cambios en el software han sido igual de importantes debido a que la capacidad para desarrollar sistemas grandes y complejos ha mejorado drásticamente. Nuestros servicios e infraestructuras nacionales —energía, comunicaciones y transporte— dependen de sistemas informáticos muy amplios, complejos y fiables. En la construcción de sistemas de software se mezclan muchas tecnologías —J2EE, .NET, EJB, SAP, BPEL4WS, SOAP, CBSE— que permiten que aplicaciones grandes basadas en Web sean desarrolladas mucho más rápido.
Las organizaciones profesionales internacionales que se han preocupado por los actuales métodos y técnicas de ingeniería del software más destacadas son ACM (Association of Computing Machinery) e IEEE (Institute for Electrical and Electronics Enginners) y de las cuales la primera, es de donde las mallas curriculares están absorbiendo su estructura según las carreras de ingeniería. Es así como la Universidad Técnica de Machala (UTMACH) con la carrera de Ingeniería de Sistemas se encuentra en un rediseño de su titulación modificando su nomenclatura de Ingeniería de Sistemas a Ingeniería de Tecnologías de Información basado en ACM; integrando de esta manera diferentes temas comprendidos también en el área de la Ingeniería de Software.
Un tema de impacto en el libro son los fundamentos de ingeniería de software; es decir la adquisición de conocimiento claro a interrogantes que dan inicio al proceso de desarrollo de un sistema como es el tener identificado sus costos, responsabilidades, conocer los requerimientos de los interesados; en que los fallos de funcionamiento tienen resultados nefastos y donde la seguridad del sistema es crítica.

En este libro mostraremos algunas pruebas o test que se realizan a estudiantes, ejercicios y proyectos diseñados por sus autores; esperamos que el texto les resulte interesante a los ingenieros de software, ingenieros de sistemas y estudiantes de ingeniería.

Estructura del libro

La estructura del libro está basada en asignaturas que se desarrollan en la UTMACH como Análisis Orientado a Objetos o Ingeniería de Software I (fundamentos de la ingeniería del software). Está organizado de seis unidades.

Unidad 1: Fundamentos y Ciclo de Vida de Software.
Unidad 2: Estudio Preliminar y Gestión del Proyecto.
Unidad 3: Ingeniería de Requerimientos.
Unidad 4: Paradigma Orientado a Objetos con UML
Unidad 5: Modelo Conceptual
Unidad 6: Comportamiento de los Sistemas

Guía para la lectura

Este libro está orientado a estudiantes de ingeniería de sistemas, ingenieros de software, graduados. Puede ser usado en cursos generales de ingeniería del software o en cursos específicos tal como se da el caso en la UTMACH, en donde se dicta en 5to semestre, como Ingeniería de Software I, y en otras carreras de Ingeniería como Análisis Orientado a Objetos. A los ingenieros de sistemas que trabajan en la organización, este libro puede serles útil como lectura general y como actualización de sus saberes en temas específicos como ingeniería de requerimientos. Los ejemplos en el libro son artículos de los autores sobre metodologías a utilizar en diferentes aplicaciones que los ingenieros deben resolver.

Usando libro para enseñar

Este libro se ha realizado para que pueda ser utilizado en:
Módulos de introducción a la ingeniería del software: Orientado para estudiantes que no tienen conocimiento en ingeniería del software. Se debe comenzar con la parte introductoria relacionada a la Ingeniería. Esto ayudará a los estudiantes a tener una visión general de la materia con la oportunidad de llevar a cabo un estudio más explícito por parte de los alumnos interesados. Si el módulo está basado en proyectos de software, las unidades de libro facilitan suficiente material para una adecuada especificación de requisitos de software.

Índice

Ingeniería de Software..... **¡Error! Marcador no definido.**
Introducción al Capítulo I ..... **¡Error! Marcador no definido.**
1 Fundamentos y Ciclo de Vida de Software .. **¡Error! Marcador no definido.**
1.1 Sistemas de Información ..... **¡Error! Marcador no definido.**
1.1.1 Sistemas ..... **¡Error! Marcador no definido.**
1.1.2 Sistemas de Información..... **¡Error! Marcador no definido.**
1.1.3 ¿Qué es un buen sistema? ..... **¡Error! Marcador no definido.**
1.1.4 ¿Se tiene buenos sistemas?..... **¡Error! Marcador no definido.**
1.1.5 ¿Cómo son los buenos sistemas? ... **¡Error! Marcador no definido.**
1.1.6 ¿Tipos de sistemas de acuerdo a las necesidades del negocio?**¡Error! Marcador**
1.1.7 ¿Qué es un proceso de software? ... **¡Error! Marcador no definido.**
1.2 Ciclo de Vida del Software ..... **¡Error! Marcador no definido.**
1.2.1 ¿Qué es el análisis de sistemas y cuál es la función del analista de sistemas?..... **¡Error! Marcador no definido.**
1.2.2 Ciclo de Vida de Desarrollo de Sistemas**¡Error! Marcador no definido.**
1.2.3 Modelos de Desarrollo..... **¡Error! Marcador no definido.**
1.2.4 ¿Qué es una metodología de Desarrollo de Software?**¡Error! Marcador no defini**
TALLER 34
EVALUACIÓN 36
GLOSARIO..... 40
SOLUCIONARIO (EVALUACIÓN) 42
Introducción al Capítulo II 44
2 Estudio Preliminar y Gestión de Proyecto de Software 45
2.1 Gestión de Proyectos de Software 46
2.1.1 Actividades de Gestión 47
2.1.2 Planificación del Proyecto..... 48
2.1.3 El Plan del Proyecto 49
2.1.4 Hitos y Entregas 50

2.1.5 Calendarización del Proyecto 51
2.2 Problemas y Oportunidades 51
2.2.1 Analizar las áreas de empresa 52
2.2.2 Técnicas de recolección de datos 52
2.2.3 Matriz de Problemas y Oportunidades 53
2.2.4 Ámbito del Software 54
2.3 Estudio de Factibilidad 55
2.3.1 Factibilidad Operativa 55
2.3.2 Factibilidad Técnica 56
2.3.3 Factibilidad Económica 56
2.4 Análisis de la Factibilidad Económica 56
2.4.1 Presupuesto 56
2.4.2 Matriz de Costo - Beneficio 57
2.4.3 Amortización y Rentabilidad 57
TALLER 58
EVALUACIÓN 61
GLOSARIO 65
SOLUCIONARIO (EVALUACIÓN) 65
Introducción al Capítulo III 68
3 Ingeniería de Requerimientos 68
3.1 Especificación de Requisitos 68
3.1.1 Requerimientos Funcionales y No Funcionales 69
..... 70
3.1.2 Requerimientos del Dominio 72
3.1.3 Requerimientos del Usuario 72
3.1.4 Requerimientos del Sistema 73
3.1.5 Obtención y análisis de requerimientos 73
3.1.6 Perfil de Usuario 74
3.2 Estándar IEEE 74
3.2.1 Documento de Requerimientos de Software estándar IEEE 830-1998 77
TALLER 79
EVALUACIÓN 83

GLOSARIO 86
SOLUCIONARIO 87
Introducción al Capítulo IV 89
4 Paradigma Orientado a Objetos con UML 89
4.1 Paradigmas Orientados a Objetos y Artefactos UML 90
4.1.1 Paradigma Orientado a Objetos 90
4.1.2 Herramientas CASE 92
4.1.3 ¿Qué es UML? 93
4.1.4 Repertorio de artefactos de modelado UML 2.0 94
4.2 Diagramas de Casos de Uso 94
4.2.1 Casos de Uso – Importancia 94
4.2.2 Elementos Básicos: Actor, Caso de Uso 95
4.2.3 Descripción de un caso de Uso – Notación Gráfica 96
4.2.4 Caso de Uso y Escenarios Organización 98
4.3 Formatos de Diagramas de Casos de Uso 99
4.3.1 Diagramas de Caso de Uso - Elementos 99
4.3.2 Formato de Alto Nivel 101
4.3.3 Formato Expandido 101
4.3.4 Matriz de Trazabilidad 102
4.4 Diagramas de Actividad 103
4.4.1 Software de modelado 103
4.4.2 Diagramas de Actividad 103
TALLER 106
EVALUACIÓN 112
GLOSARIO 115
SOLUCIONARIO (EVALUACIÓN) 117
Introducción al Capítulo V 119
5 Modelo Conceptual 119
5.1 Diagrama Conceptual 120
5.1.1 Definición 120
5.1.2 ¿Qué nos muestra el Modelo Conceptual? 120
5.1.3 Estrategias de Desarrollo 121

5.1.4 Asociaciones 121
5.1.5 Multiplicidad 122
5.1.6 Diagrama Conceptual 123
TALLER 124
EVALUACIÓN 126
GLOSARIO 130
SOLUCIONARIO (EVALUACIÓN) 130
..... 131
Introducción al Capítulo VI 132
6 Comportamiento de los Sistemas 132
6.1 Definición 133
6.1.1 Actividades y Dependencias 133
6.1.2 Eventos y Operaciones 134
6.1.3 ¿Cómo elaborar un diagrama de secuencia? 135
6.2 Contratos de las Operaciones 135
6.2.1 Elementos 136
6.2.2 Actividades y Dependencias de los Contratos 136
6.2.3 Secciones de un Contrato 137
6.2.4 Post condiciones - Precondiciones 137
TALLER 139
EVALUACIÓN 142
GLOSARIO 146
SOLUCIONARIO (EVALUACIÓN) 147
Bibliografía 148

Introducción

Los sistemas automatizados hoy en día tienen una extraordinaria acogida comparada a épocas anteriores, y todo esto los ha llevado a enfrentarse con la época actual. Sin embargo, existen muchos problemas en los proyectos de software, como por ejemplo retrasos en la entrega y lo más importante problemas de calidad.

Esto ha hecho que la ingeniería de software difunda una serie de esquemas para evaluar y mejorar la calidad en muchos aspectos y se busquen procesos de mejora la cual los lleva a plantear procesos, metodologías y mejorar estándares de calidad.

Es por ello, que existen muchas herramientas automatizadas que han salido a flote con el único objetivo de ayudar a precisar y emplear un proceso de desarrollo de software seguro.

Tal vez suene erróneo, pero a pesar de todas las metodologías existentes, aún existen procesos de desarrollo informales, parciales y muchos de los casos no son confiables.

La Ingeniería de Requerimientos (IR) efectúa un papel fundamental en el transcurso de la producción de software, esta se orienta en un área fundamental: la definición de lo que se desea lograr. Su primordial labor reside en la generación de especificaciones correctas que representen con claridad, sin imprecisiones, en forma consistente y comprimida, el comportamiento del sistema. De esta manera, se pretende minimizar los problemas relacionados al desarrollo de sistemas.

Ingeniería de software

Fundamentos y ciclos de la vida del software

Competencias

- Define conceptos de sistemas de información.
- Reconoce sistemas de acuerdo a los atributos y principios del software.
- Reconoce sistemas de acuerdo a los atributos y principios del software.

Resultados de aprendizaje:

Identificar los distintos conceptos de sistemas de información en aplicaciones concretas y ser capaz de aplicar los conceptos teóricos en distintos casos de la realidad;
Identificar, clasificar y describir los tipos de sistemas de Información.
Identificar el ciclo de vida apropiado según las necesidades del Cliente.

Contenidos

1.1 Sistemas de Información.
1.2 Ciclo de Vida del Software

Introducción al capítulo I

Este capítulo pretende ser una guía acerca de los temas relevantes de la ingeniería de software, tal es el caso de los fundamentos y ciclo de vida de software, en la cual se detallará las definiciones, conceptos, y ejemplos; esto permitirá comprender mejor los temas y subtemas expuestos dentro de este documento.
Otro de los aspectos importante de la ingeniería de software, es el ciclo de vida de un software, en la cual también comprenderá definiciones de varios autores, acerca de los modelos y metodologías que existen en la actualidad, comprender acerca de los tipos de metodologías existentes, tal como son agiles y tradicionales. Es importante conocer estos aspectos, ya que son un punto de partida para todo desarrollo de proyectos de sistemas, conocer aspectos generales tales como el problema, asociarse con el ambiente del trabajo, entre otros.
El objetivo de este capítulo es dar a conocer los fundamentos necesarios de la ingeniería de software, conociendo puntos importantes de los temas ya anteriormente mencionados, para llevar al éxito el desarrollo de proyectos de software.

1 Fundamentos y ciclos de la vida del software

Para centrarnos cuanto antes en el objeto de la Unidad 1, Fundamentos y Ciclo de Vida de Software, estableceremos definiciones sobre sistema de información (SI) y cuáles son sus componentes esenciales, ya que conocer estos aspectos nos permite ir comprendiendo ciertos conceptos relacionados con el ciclo de vida del software. Seguidamente se explicará el ciclo de vida de software, sus modelos de desarrollo y metodologías, así como de la implementación, tomando en cuentas las características de los modelos y metodologías, sin olvidar que es importante cumplir las exigencias y expectativas de los clientes.

1.1 Sistemas de información

A continuación se detallarán conceptos y aspectos importantes acerca de los sistemas de información, para lograr entender la importancia de estos. Entre los varios aspectos a conocer están:

1.1.1 Sistemas

Se puede definir al sistema como un conjunto de varios elementos que se relacionan entre sí, y logran un objetivo por medio de la realización de actividades, la cuales operan con elementos de entradas para lograr salidas de nuevos elementos o resultados.

El diccionario de la Real Academia Española (RAE) define el vocablo sistema como "un conjunto de cosas que ordenadamente relacionadas entre sí contribuyen a un determinado objetivo" [1] y el libro de Informática y comunicaciones en la empresa, por sistema entiende "un conjunto de elementos en interacción dinámica organizados para la consecución de un objetivo" [2].
Otros autores, como es el caso de Vicenç Fernández Alarcón, en el libro de desarrollo de Sistemas de Información, define al SI como "un conjunto de componentes que interaccionan entre sí para lograr un objetivo en común." [3]; Además dicho autor agrega que "existe una variedad de sistemas, la mayoría de ellos pueden representarse a través de un modelo formado por cinco bloques básicos: elementos de entrada, elemento de salida, sección de transformación, mecanismos de control y objetivos."

1.1.2 Sistemas de información

A partir de las definiciones anteriores (de sistemas), podemos establecer que un sistema de información es aquél que permite recopilar, organizar y manipular un conjunto de información necesaria, para que los integrantes de una organización puedan tomar decisiones certeras.

Entonces se puede decir que un sistema de información es un conjunto de componentes enfocados a la manipulación y administración de la información, los cuales están organizados y listos para su utilización.

Pressman Roger, Ph.D. [1992] define un sistema basado en computadora como:

"Un conjunto u ordenación de elementos organizados para llevar a cabo algún método, procedimiento o control mediante el procesamiento de información" [4]

Dar una definición exacta de lo que es un Sistemas de información, no es fácil, ya que no existe un definición en concreta. Actualmente la expresión "Sistemas de Información" es utilizada de manera común en

las organizaciones. A pesar de aquello existen autores que definen como un "conjunto de componentes relacionados entre sí, para lograr un objetivo común, el cual es cumplir con las necesidades de información dentro de una organización".

En el libro de informática y comunicaciones en la empresa, se define al sistema de información como:

"Un conjunto de recursos técnicos, humanos y económicos, interrelacionados dinámicamente, y organizados en torno al objetivo de satisfacer las necesidades de información de una organización empresarial para la gestión y la correcta adopción de decisiones." [2]

En la actualidad, el entorno que nos rodea está lleno de ejemplos de sistemas, tales como: el sistema de matriculación vehicular de Agencia Nacional de Transito (ANT), el de registro de títulos en el Ecuador (Portal Web de la Educación Superior), el de matriculación de la Universidad Técnica de Machala (SIUTMACH). En el caso del SIUTMACH, el estudiante accede y puede matricularse en las asignaturas pertinentes (mecanismos de control), verificar sus notas, actualizar sus datos personales (entrada de información), teniendo como objetivo del sistema optimizar los procesos organizacionales.
Otro de los puntos importante a conocer, es acerca de los elementos fundamentales que forman a los SI, a continuación citaremos los elementos que aparecen el libro de Informática y comunicaciones en la empresa [2]:

- La información: Lo capturado, almacenado, procesado y distribuido por el sistema. Se puede decir que es la parte fundamental tanto para la empresa como para el sistema
- Las personas: introducen y analizan la información del sistema usando la información, tecnología y otros recursos de la institución.
- Los equipos de tratamiento de la información e interacción con los usuarios: Se encarga de almacenar y procesar la información realizada por los usuarios, estos llegan a ser el hardware, software y redes de comunicaciones
- Las normas y/o técnicas de trabajo: métodos utilizados por las personas y las tecnologías para desarrollar sus habilidades.

1.1.3 ¿Qué es un buen sistema?

Un buen sistema involucra tanto conocimientos como metodologías aplicadas al desarrollo, implementación y mejora de estructuras para la resolución de problemas que se dan a menudo en la sociedad, en la cual abarca la necesidad del desarrollo de los sistemas de información.

Según el libro de Contabilidad, en su Quinta Edición, cita que todo buen sistema *"-ya sea manual o computarizado- incluye control, compatibilidad, flexibilidad y relación de costo sobre beneficio favorable"* [5].
Un buen sistema debe tener tiempo de respuesta pertinente (eficiente), fácil de utilizar (interfaz de usuario apropiada), evolución a cambios en el entorno (mantenibilidad) y ser fiable (seguro) son las características principales.

1.1.4 ¿Se tiene buenos sistemas?

Debido a la creciente diversidad de hardware y software, las demandas para reducir los tiempos de entrega de sistemas y desarrollo de software fiable se establece que los sistemas son buenos cuando cumplen la demanda del cliente en el momento (cumple parámetros de calidad); puesto que los cambios siempre existen o los procesos para los cuales fue diseñado el sistema han variado.
Para mejorar los sistemas se debe tener en cuenta las estimaciones desde el inicio del proyecto (tiempo, tamaño, defectos, etc.), un repositorio de datos de los proyectos y conocer los costos asociados.
Con lo mencionado, un buen sistema se obtiene verificando siempre las herramientas con las que se cuenta, procesos que se desarrollan con claridad y cumpliendo siempre la planificación con su respectiva calidad.

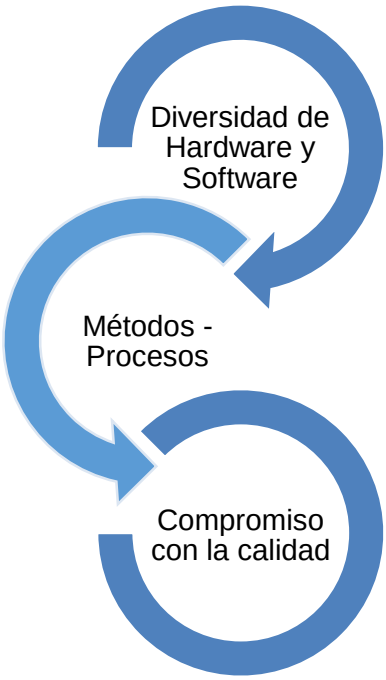


Ilustración 1: Buenos Sistemas - Ingeniería de Software

1.1.5 ¿Cómo son los buenos sistemas?

Un sistema de información será eficaz, según el libro de Informática y comunicaciones en la empresa, "si facilita la información necesaria para la organización, y será eficiente si lo realiza con los menores recursos tecnológicos, humanos y económicos posibles, y en el momento oportuno" [2]

El libro Manual para la elaboración de un sistema de estadísticas sobre justicia penal, escrito por United Nation, cita acerca del sistema de información operacional, que los "buenos sistemas operacionales, es decir, los que satisfacen las necesidades administrativas, no contengan los tipos y la cantidad de información necesarios para producir buenas estadísticas" [6]

Los buenos sistemas deben ser completos, eficientes, cumpliendo todos los requisitos del usuario y/o el cliente, es por ello que, el desarrollo de los sistemas, se rigen bajo una plantilla, modelo o metodología para el dicho desarrollo, lo cual ayuda a mejorar los procesos y por ende a formar buenos sistemas. Todo buen sistema deber ser muy eficiente y eficaz, para lograr los objetivos trazados.

1.1.4 ¿Tipos de sistemas de acuerdo a las necesidades del negocio?

Según el Libro Análisis y Diseño de Sistemas de Kendall & Kendall en su octava edición indica que los tipos de sistema son:

- TPS - Transaction Processing Systems (Sistema de procesamiento de transacciones)
- Recolecta, almacena, modifica y recupera toda la información generada por las transacciones producidas en una organización.
- Estos sistemas permiten a la organización mejorar y mantener un seguimiento o registro de sus operaciones o transacciones rutinarias, cuyos datos son almacenadas en una base de datos. Es por esta razón que también se les llama sistemas de procesamiento de datos.
- OAS - Office Automation Systems (Sistema de automatización de oficinas)
- Funcionan en la automatización de la oficina.
- Consisten en aplicaciones destinadas a ayudar al trabajo diario del administrativo de una organización, forman parte de este tipo de software los procesadores de textos, las hojas de cálculo, los editores de presentaciones, los clientes de correo electrónico, etc.
- KWS - Knowledge Work Systems (Sistemas de Trabajo con Conocimientos)
- Funcionan en el trabajo del conocimiento.
- Sistemas de información que apoyan a los trabajadores del conocimiento en la creación e integración del nuevo conocimiento en una organización.
- MIS - Management Information Systems (Sistema de información gerencial)
- Son una colección de sistemas de información que interactúan entre sí y que proporcionan información tanto para las necesidades de las operaciones como de la administración
- DSS - Decision Support Systems (Sistema de apoyo de decisiones)
- Los DSS son herramientas de mucha utilidad en Inteligencia empresarial, permiten realizar el análisis de las diferentes variables de negocio para apoyar el proceso de toma de decisiones de los directivos.
- Ayuda a los gerentes a tomar decisiones para tratar de resolver problemas semiestructurados.

- ESS - Executive Support Systems (Sistemas de Apoyo a Ejecutivos)
- Se trata de sistemas de información dirigidos a la alta dirección puesto que permiten automatizar la obtención de datos más relevantes y estratégicos para la organización, presentándolos de la forma más comprensible posible con la finalidad de poder dar seguimiento a los factores críticos de éxito.

Los sistemas de información se desarrollan con diversos propósitos, según las necesidades del negocio, tenemos:

- Sistemas de Controles Embebidos: sistemas que controlan y manejan dispositivos de hardware. Por ejemplo: software para teléfonos celulares, anti-bloqueo de frenos, microondas, etc.

- Sistemas de procesamiento por lotes: sistemas diseñados para procesar datos en grandes volúmenes. Por ejemplo: sistemas de facturación.
- Sistemas de entretenimiento: sistemas principalmente de uso personal y orientado a entretener al usuario.
- Sistemas para modelamiento y simulación: sistemas para científicos e ingenieros para modelar procesos físicos o experimentos. Usualmente requieren altos niveles de desempeño.
- Sistemas de recolección de datos: colectan datos desde su entorno por medio de un conjunto de sensores y usualmente se conectan dentro de maquinarias o en ubicaciones remotas.

1.1.5 ¿Qué es un proceso de software?

Es un conjunto de elementos y actividades, los cuales ayudan a la implementación de los sistemas de software.

En el libro de Ingeniería del software, escrito por Ian Sommerville, define al proceso de software como "un conjunto de actividades que conducen a la creación de un producto software", en la cual "dichas actividades pueden consistir en el desarrollo de software desde cero en un lenguaje de programación estándar como Java o C." [7]

Los procesos de software no son del todo fáciles, ya que la elaboración de un sistema de software requiere de la toma de decisiones de personas relacionadas al mismo, normalmente con puntos de vista y

objetivos distintos; debido a esto los procesos de automatización han sido complejos de implementar. Es por ello que existen ayudas para mejorar los procesos de automatización, ente los que se destacan el uso de las herramientas CASE, estas pueden ayudar a mejorar varias de las actividades del proceso a automatizar.

1.2 Ciclo de vida del software

Como ya se mencionó Ian Sommerville, (Séptima Edición), Libro de Ingeniería de Software define al proceso de Software, como:

"*Un conjunto de actividades cuya meta es el desarrollo o evolución del software*" [7].

El proceso de software, también se lo conoce como ciclo de vida del desarrollo de software, el mismo es una estructura al desarrollo de un producto (software).

1.2.1 ¿Qué es el análisis de sistemas y cuál es la función del analista de sistemas?

Según el libro Análisis y diseño de sistemas, escrito por Julie E. Kendall cita lo siguiente:
"La siguiente fase que enfrenta el analista es la determinación de los requerimientos de información de los usuarios" [8]**,** además agrega que para realizar la toma de estos requerimientos, se utilizan herramientas tales como *"entrevistas, los muestreos, la investigación de datos impresos y la aplicación de cuestionarios"*

El analista de sistemas es el encargado de documentar las necesidades o requerimientos del software a desarrollar (tomadas el cliente y/o usuarios), debe poseer facilidades en el manejo de personas y en el estudio de soluciones factibles a posibles problemas; además de ser auto disciplinado y auto motivado como individuo.
Según el Libro Análisis y Diseño de Sistemas de Kendall & Kendall en su octava edición indica que el analista de sistema:
"El analista de sistema evalúa en forma sistemática cómo interactúan los usuarios con la tecnología y cómo operan las empresas, para lo cual examina los procesos de entrada/salida de los datos y la producción de información con la intención de mejorar los procesos organizacionales" [8]

En el medio, un analista de sistema debe ser capaz de trabajar con personas de todo tipo para recopilar información adecuada sobre el software a desarrollar.
Es importante que el analista de sistema se involucre con el ambiente de la empresa, esto permite al analista, conocer de una manera más profunda y concreta, acerca de los procesos que se dan en la misma, ya sea procesos de entrada y salida de los datos, ya que esto permite mejorar los procesos en la implementación.

1.2.2 ¿Ciclo de vida de desarrollo de sistemas?

Se entiende por ciclo de vida del desarrollo de sistemas a un conjunto estructurado de actividades y resultados asociados requeridos para desarrollar un software.
El ciclo de vida de desarrollo de sistemas en la actualidad se enfoca en la metodología de desarrollo seleccionada para documentar paso a paso la elaboración del software.
Según *Gonzalo Méndez (2008-2009),* Universidad Complutense de Madrid – Dpto. de Ingeniería de Software e Inteligencia Artificial, la estructura se compone de:

- Especificación: Establecer requisitos y restricciones, esta fase identifica las necesidades del cliente a nivel funcional y no funcional, de forma que se refleje en detalle las funcionalidades a implementar.
- Diseño: Producir un modelo de papel del sistema, el diseño de un sistema de información produce los detalles que establecen la forma en la que el sistema cumplirá con los requerimientos identificados durante la fase de análisis. Los especialistas en sistemas se refieren, con frecuencia, a esta etapa como diseño lógico en contraste con la del desarrollo del software, a la que denominan diseño físico.
- Implementación: Construcción del sistema de software.
- Esta etapa es donde se desarrolla el software definido en la especificación de requisitos, descrito en la fase de análisis y detallado en el diseño.
- Validación: Verificar que el sistema cumple con las especificaciones requeridas. Esto se da una vez que el sistema pasa la etapa de pruebas, es donde el cliente debe verificar que el sistema cumple las especificaciones de requisitos que describen sus necesidades.
- Instalación: Una vez concluidas las etapas de desarrollo de un sistema de información llega el instante de que poner el

sistema en funcionamiento, su instalación. Esta es la fase en donde se entrega el sistema al usuario.

- Evolución y mantenimiento: Cambiar/adaptar el software según las demandas; reparar fallos en el sistema. El mantenimiento del producto exige rehacer parte del trabajo inicial, que puede corresponder a cualquiera de las actividades de las etapas anteriores.

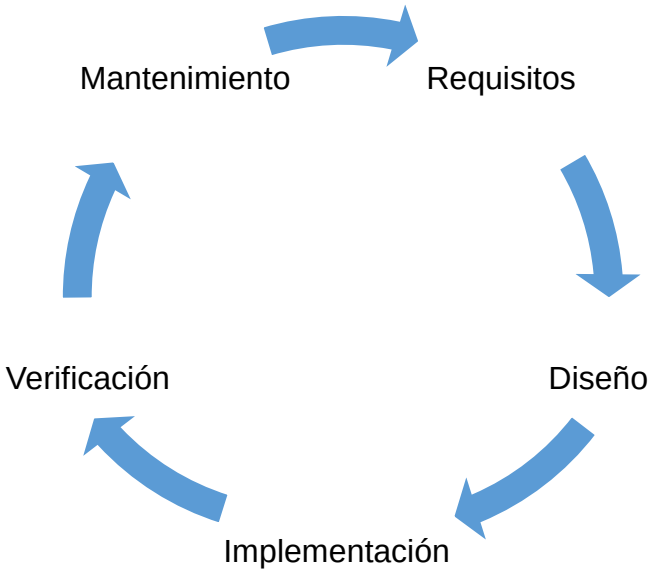


Ilustración 2: Ciclo de Vida Software

1.2.3 ¿Modelos de desarrollo?

Es un conjunto de actividades y estrategias que permiten desarrollar y mantener el software de forma más precisa, consistente y estructurada.
Una pregunta que siempre nos realizamos es: ¿Por qué debemos saber esto? Necesitamos saber cómo se desarrolla el software, prever de antemano problemas que pudieran presentarse en el desarrollo del sistema que deseamos producir. Cuando investigamos esta temática los artículos hablan de modelos (línea base, vocabulario y conocimientos) guía y buenas prácticas.
Entre los modelos de proceso de software tenemos:

Cascada: Según el libro de ingeniería de software, escrito por Ian Sommerville, “las actividades fundamentales del proceso de especificación, desarrollo, validación y evolución, y los representa como fases separadas del proceso, tales como la especificación de

requerimiento, el diseño del software, la implementación, las pruebas, etcétera." [7]
En el modelo de cascada el proceso de desarrollo del software, el cual abarca fases tales como especificación, desarrollo, validación y otros, se realiza de manera ordenada, dichos procesos como: el de especificación de requerimientos, diseño del sistema, implementación y otros se llevan a cabo consecutivamente, es decir no pueden ejecutarse sin que haya finalizado satisfactoriamente el proceso anterior.

Entre las características de este modelo están:

- La etapa previa debe finalizar antes de iniciar la siguiente.
- Se producen y aprueban documentos al final de cada etapa
- Es el modelo clásico – de sentido común.
- Ampliamente conocido, comprendido y usado.

Tabla 1: Modelo Cascada

Ventajas	Desventajas
Fácil de manejar – cada fase tiene entregables específicos.	Es difícil regresar a una fase previa.
Refuerza buenos hábitos: Definir requerimientos antes de diseñar.	El software es entregado al final del ciclo de vida del proyecto.
Evita desperdiciar el tiempo implementando un sistema no deseado. Usado en proyectos estables, con requerimientos claramente definidos. (Proyectos pequeños).	No es apropiado para proyectos complejos o aquellos en los que los requerimientos tienen alta probabilidad de cambio.

Según el libro de ingeniería de software [7], establece que las etapas que tiene el modelo cascada son:
Análisis y definición de requerimientos, diseño del sistema y del software, implementación y pruebas de unidades, integración y pruebas del sistema, funcionamiento y mantenimiento.

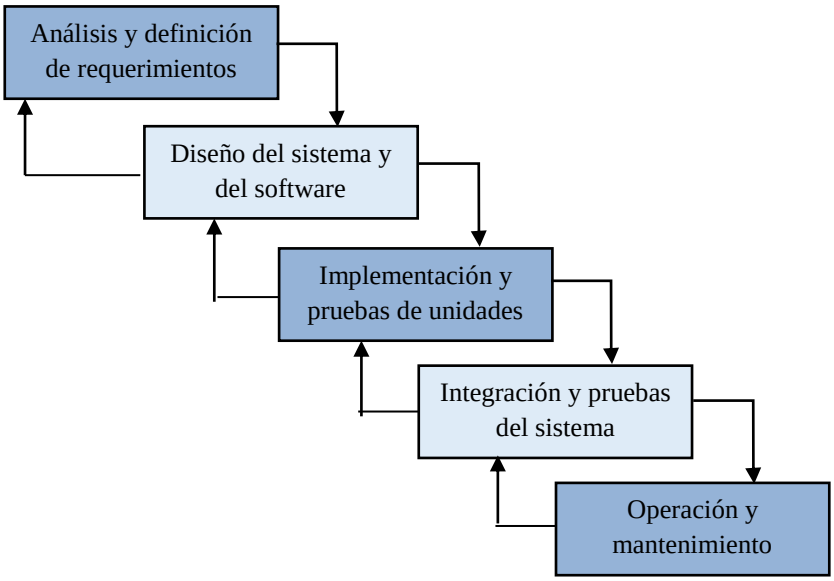


Ilustración 3: Ciclo de Vida del modelo cascada

Orientado al reúso – Basado en Componentes:

En el libro de Informáticos Generalitat Valenciana Grupos a Y B. Temario Bloque Específico Volumen I, cita que este modelo de desarrollo "la nueva aplicación se construyen reusando los componentes y aplicaciones ya existentes.", lo que ayudaría a desarrollar de una manera más rápida al programador, ya que "los programadores escriben solo el código necesario para conectar los componentes." [9]

Este modelo básicamente se sustenta en la reutilización de los componentes que pueden formar el sistema. Esto ayuda a evitar desarrollarlos desde cero, sino en una integración de los componentes necesarios que forman al sistema.

Entre las características están:

- Servicios Web, los mismos que son desarrollados de acuerdo a los estándares de servicios y están disponibles por invocación remota.
- Colecciones de objetos que son desarrollados como un paquete
- Sistemas de software stand – alone que pueden ser configurados para su uso en un ambiente en particular

Tabla 2: Modelo Basado en Componentes

Ventajas	Desventajas
Reduce la cantidad de software que debe ser desarrollado lo que reduce costos y riesgos	Se pierde el control sobre la evolución del software si las nuevas versiones de los componentes reusables no están bajo el control de la organización
Usualmente se disminuyen los tiempos de entrega del software.	El ajuste de requerimientos podría producir un sistema que no satisface las necesidades reales del usuario.

Iterativo:

Según el libro de Curso de Ingeniería de Software, cita que "los modelos incrementales incluyen los modelos de desarrollo cuya constitución es de pequeños ciclos de desarrollo iterativo, es decir, a cada ciclo nuevos incrementos son añadidos en el software (Pressman, 2006, p.40), que gana funcionalidades durante el proyecto." [10]; Además en este modelo, según el libro de Ingeniería de proyectos informáticos: actividades y procedimientos, escrito por José Sánchez, "se encuentra la flexibilidad para acomodarse a nuevos requisitos o cambio tácticos en los objetivos de negocio." [11]

Entre las características de este modelo están:

- Desarrollo de un sistema por repetición de ciclos.
- Divide el desarrollo de software de una aplicación grande en pedazos pequeños.
- Iteración: nuevas características son diseñadas, desarrolladas, probadas y puestas en producción.

El modelo Iterativo se subdivide en Desarrollo Incremental, Desarrollo Ágil entre otros.

El Desarrollo Incremental:

- El sistema es entregado en partes (incrementos / módulos).
- El cliente debe de clasificar las partes de acuerdo a su importancia (prioridad alta, media, baja).
- El incremento de más alta prioridad se entrega primero.
- Cada incremento se pone en producción.

Tabla 3: Modelo Iterativo - Desarrollo Incremental

Ventajas	Desventajas
No hay que esperar hasta el final para tener un sistema en producción.	Para los usuarios es difícil esperar hasta que su funcionalidad esté lista (reemplazo de sistema)
Las funcionalidades más importantes se desarrollan primero (se prueban más).	A veces se complica identificar servicios comunes a todos los incrementos (el enfoque es por incremento).
Se van descubriendo cambios en requerimientos de los incrementos no desarrollados (fácil incorporar cambios). Es más fácil acostumbrarse al nuevo sistema.	Impacto en el proceso de contratación (los requerimientos pueden cambiar a medida que se desarrollan los incrementos).

- No se debe de usar cuando los sistemas son muy grandes con equipos trabajando en diferentes localidades.
- No se debe usar en sistemas embebidos: todos los requerimientos deben ser analizados (interacciones Hardware – Software) para no comprometer la seguridad de los usuarios.

El Desarrollo Ágil:

- Incrementos (agrega características al producto) desarrollados en ciclos cortos.
- Cada pequeño entregable es construido sobre los anteriores entregables.
- Se prueba cuidadosamente el incremento generado.

Tabla 4: Modelo Iterativo - Desarrollo Ágil

Ventajas	Desventajas
Promueve trabajo en equipo.	Mayor riesgo (mantenibilidad y extensibilidad)
Documentación mínima.	Depende fuertemente en la interacción con el cliente.
Poca planificación.	Alta dependencia de los individuos ya que se genera poca

	documentación del proyecto.
Funcionalidades pueden ser desarrolladas rápidamente.	Es un reto transferir conocimientos a nuevos miembros del equipo debido a la falta de documentación

Un modelo de proceso, o paradigma de Ingeniería de Software, es una plantilla, patrón o marco que define el proceso a través del cual se crea software. Dicho de otra forma, los procesos son instancias de un modelo de proceso.

1.2.4 ¿Qué es una metodología de desarrollo de software?

Según EDUCACION Y TECNOLOGIA: Un binomio excepcional, cita que las metodologías de desarrollo "influyen directamente en el proceso de construcción y se elaboran a partir del marco definido por uno o más ciclos de vida." [12], en el mismo libro define q las metodologías son "un conjunto de pasos o procedimientos que deben seguirse para desarrollo del software"

Los modelos de desarrollo de software son una representación abstracta de una manera en particular. No representan cómo se debe desarrollar el software, sino dan un enfoque de organización que podría variar según la naturaleza de cada proyecto, los métodos y herramientas a utilizar, los controles y entregas requeridos. Cada modelo cuenta con sus pro y contra por ello se debería escoger el más apropiado para el proyecto, considerando las necesidades.
Un proceso de software especificado de manera completa suele denominarse "Metodología". Las metodologías como ya se mencionaron se basan en modelo de desarrollo; adicionalmente se debería definir con precisión los artefactos, roles y actividades involucradas, junto con prácticas y técnicas recomendadas, etc.

Conceptos importantes

Análisis de sistema: es la recolección de datos e información, para determinar requisitos para el desarrollo del sistema.

Analista de sistemas: es la persona que tiene como objetivo el analizar un determinado problema, y dar solución mediante la implementación de un sistema de información.

Ciclo de vida de software: describe los diferentes pasos o seguimientos para el desarrollo de software, mediante una fase inicial hasta una fase

final, logrando un proceso evolutivo y ordenado, logrando obtener un producto final.

Modelos de desarrollo: Es un conjunto de actividades y estrategias que ayudan a desarrollar y mantener el software de una manera más precisa, consistente y estructurada.

Procesos de software: Es un conjunto de elementos, tales como, personas, reglas, tecnologías, actividades, artefactos, entre otros, los cuales son necesarios para desarrollar, instalar y dar mantenimiento a un producto de software.

Sistema de información: es la representación de un conjunto de elementos o componentes, los mismos que se relacionan entre sí, para cumplir un objetivo, toma de decisión o mejorar el control de la organización.

Lecturas complementarias

Del articulo académico: "Modelos de Ciclo de Vida de Desarrollo de Software en el Contexto de la Industria Colombiana de Software.", de Hugo F. Arboleda Jiménez. MSc. Disponible en el siguiente enlace:

http://www.emn.fr/z-info/harbol07/ACIS.pdf

Desarrollar las siguientes preguntas:

1. ¿Cómo se caracterizan las metodologías tradicionales?

2. ¿Cuál es la característica fundamental de las metodologías agiles?

3. ¿En que se basan las metodologías de Cristal?

4. ¿Cuál fue uno de los primeros modelos de ciclo de vida, y que formalizo?

5. Defina a la metodología XP e indique que promueve.

Del artículo "Los Sistemas De Información: Evolución Y Desarrollo", de Alejandro Hernandez Trasobares. Disponible en el siguiente enlace:

https://www.google.com.ec/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwiA7bqQmJTKAhWGXhQKHckMC6gQFggaMAA&url=http%3A%2F%2Fdialnet.unirioja.es%2Fdescarga%2Farticulo%2F793097.pdf&usg=AFQjCNFJIZgZ-XpVrpuzIO4wECr-pFxcOw&sig2=etvtSfAp9peY0a5umfVW7w&bvm=bv.110151844,d.eWE

Desarrollar las siguientes preguntas:

1. Cuál es la definición de Andreu, Ricart y Valor (1991), con respecto a ¿Qué es un sistema de información?

2. Cite un objetivo que debe poseer todo sistema de información.

3. Cite la clasificación de los sistemas de información según K y J Laudon.

4. Cite las siete etapas fundamentales del proceso de desarrollo de los sistemas de información.

5. Cite una de las principales causas que originan el fracaso de los sistemas de información.

TALLER

TALLER N° 1 – Sistemas de Información

1. **Cite una definición de Sistemas.**

   _____
   _____
   _____

2. **Cite 3 ejemplos de sistemas de información.**

   _____
   _____
   _____

3. **¿Qué es un buen sistema?**

   _____
   _____
   _____

4. **¿Cómo se obtiene un buen sistema?**

   _____
   _____
   _____

5. **De 3 ejemplos de sistemas de información que se desarrollan según las necesidades del negocio.**

   _____
   _____

_____

TALLER N° 2 – Proceso de Software

1. **¿Qué es un proceso de software?**

   _____
   _____
   _____

2. **Complete:**

   El proceso de software, también se lo conoce como _____, la misma que es una estructura al desarrollo de un _____ (software).

3. **Complete:**

   El analista de sistema evalúa en forma sistemática cómo interactúan los _____ con la tecnología y cómo operan las empresas, para lo cual examina los _____ de entrada/salida de los datos y la producción de información con la intención de mejorar los procesos _____.

4. **Unir lo correcto:**

Sistemas que controlan y manejan dispositivos de hardware.	**Sistemas de Controles Embebidos**
Sistemas diseñados para procesar datos en grandes volúmenes	**Sistemas de entretenimiento**
Sistemas principalmente de uso personal y orientado a entretener al usuario.	Sistemas de procesamiento por lotes

5. **¿Qué ayuda a mejorar los procesos de software?**

   _____

TALLER N° 3 – Ciclo de Vida del Software

1. **¿Qué es el ciclo de vida del software?**

   _____
   _____

_____

2. **Complete:**

El proceso de software, también conocido como _____, la misma que es una estructura al desarrollo de un producto.

3. **Cite 3 de las herramientas utilizadas por los analistas de sistema, para la toma de requerimientos.**

_____
_____
_____

4. **¿Cuál es la estructura del ciclo de vida de desarrollo de sistemas?**

_____
_____
_____

5. **¿Qué es un modelo de desarrollo y mencione dos de ellos?**

_____
_____
_____

EVALUACIÓN

	FORMATO	**SIMPLE**
	Contexto	
	Planteamiento	**Es un conjunto de componentes que interaccionan entre sí para lograr un objetivo común para satisfacer las necesidades de una organización**
	OPCIÓN A	Sistemas de Información
	OPCIÓN B	Ingeniería de software
	OPCIÓN C	Asegurar la calidad
	OPCIÓN D	Principios de la Ingeniería de Software
	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios
2	**FORMATO**	SIMPLE

	Contexto	
	Planteamiento	**Sistemas diseñados para procesar datos en grandes volúmenes.**
	OPCIÓN A	Aplicaciones basadas en transacciones interactivas
	OPCIÓN B	Aplicaciones stand-alone
	OPCIÓN C	Sistemas de controles embebidos
	OPCIÓN D	Sistemas de procesamiento por lotes
	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios
3	**FORMATO**	SIMPLE
	Contexto	
	Planteamiento	**Sistemas para científicos e ingenieros para modelar procesos físicos o experimentos. Usualmente requieren altos niveles de desempeño**
	OPCIÓN A	Sistemas de entretenimiento
	OPCIÓN B	Sistemas para modelamiento y simulación
	OPCIÓN C	Sistemas de recolección de datos
	OPCIÓN D	Sistemas de controles embebidos
	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios
4	**FORMATO**	SIMPLE
	Contexto	
	Planteamiento	**Incrementos (agrega características al producto) desarrollados en ciclos cortos. Cada pequeño entregable es construido sobre los anteriores entregables.**
	OPCIÓN A	Desarrollo incremental
	OPCIÓN B	Modelo Iterativo
	OPCIÓN C	Modelos Cascada
	OPCIÓN D	Desarrollo Ágil
	Respuesta Correcta	
	Nivel	1

	Operación Cognitiva	Aplicación de Conceptos y Principios
5	**FORMATO**	SIMPLE
	Contexto	
	Planteamiento	**Permiten desarrollar y mantener el software de forma más precisa, consistente y estructurada.**
	OPCIÓN A	Modelo Iterativo
	OPCIÓN B	Modelos Cascada
	OPCIÓN C	Modelos de Proceso de Software
	OPCIÓN D	Desarrollo basado en componentes
	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios
6	**FORMATO**	**SIMPLE**
	Contexto	
	Planteamiento	**Las metodologías se basan en:**
	OPCIÓN A	Sistemas de Información
	OPCIÓN B	modelo de desarrollo
	OPCIÓN C	Asegurar la calidad
	OPCIÓN D	Principios de la Ingeniería de Software
	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios
7	**FORMATO**	SIMPLE
	Contexto	
	Planteamiento	**Análisis y definición de requerimientos, diseño del sistema y del software, implementación y pruebas de unidades, integración y pruebas del sistema, funcionamiento y mantenimiento. ¿Son etapas que tiene el modelo cascada?**
	OPCIÓN A	Verdadero
	OPCIÓN B	Falso
	Respuesta Correcta	

	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios
8	**FORMATO**	SIMPLE
	Contexto	
	Planteamiento	**Es un conjunto estructurado de _____ y _____ asociados requeridos para _____**
	OPCIÓN A	Sistemas – resultados – elaboración de programas.
	OPCIÓN B	Actividades – resultados – desarrollar un software.
	OPCIÓN C	Actividades – no resultados – desarrollar un software
	OPCIÓN D	Sistemas – actividades – desarrollar un software
	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios
9	**FORMATO**	SIMPLE
	Contexto	
	Planteamiento	**Sistemas para científicos e ingenieros para modelar procesos físicos o experimentos. Usualmente requieren altos niveles de desempeño.**
	OPCIÓN A	Desarrollo basado en componentes
	OPCIÓN B	Sistema de entretenimiento
	OPCIÓN C	Sistema de recolección de datos
	OPCIÓN D	Sistemas para modelamiento y simulación
	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios
10	**FORMATO**	SIMPLE
	Contexto	
	Planteamiento	**El analista de sistema evalúa en forma sistemática cómo interactúan los usuarios con la tecnología y cómo operan las empresas, para**

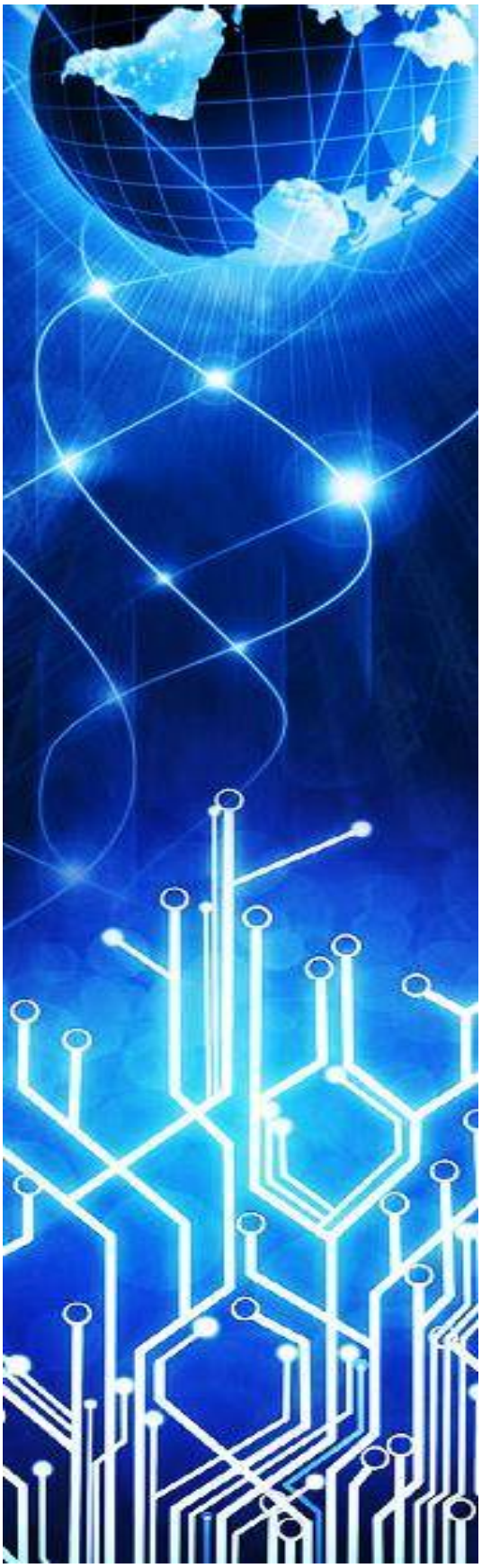
		lo cual examina los procesos de entrada/salida de los datos y la producción de información con la intención de mejorar los procesos organizacionales
	OPCIÓN A	Verdadero
	OPCIÓN B	Falso
	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios

GLOSARIO

- **Análisis**: es el estudio de un hecho, o diagnóstico de un problema, para mejorar el desarrollo de un sistema.
- **Ciclo**: es una serie de etapas, estados o actos por las que pasa un hecho o elemento, logrando cumplir un determinado objetivo.
- **Desarrollo:** es un proceso o evolución por el cual se logra cumplir tareas específicas.
- **Información**: es un conjunto de datos estructurados con respecto a algún objeto o tema.
- **Metodología**: es una plantilla, o guía por el cual se fija un proceso o procesos para lograr un objetivo.
- **Modelo**: es la representación de las partes de un sistema.
- **Proceso**: es un conjunto de pasos o instrucciones, las misma que se realizan de manera ordenada.
- **Sistema**: es un conjunto de elementos o partes que tienen una relación entre sí para lograr un objetivo.
- **Software**: es un o unos programas que pueden ser ejecutado desde un ordenador, para cumplir ciertas tareas.
- **Requerimiento**: es una petición, necesidad o condición para resolver un determinado problema.

SOLUCIONARIO (EVALUACIÓN)

1. A) Sistemas de Información
2. D) Sistemas de procesamiento por lotes
3. C) sistemas que controlan y manejan dispositivos de hardware
4. D) desarrollo Ágil
5. C) Modelos de Proceso de Software
6. B) modelo de desarrollo
7. A) Verdadero
8. B) Actividades – resultados – desarrollar un software.
9. D) Sistemas para modelamiento y simulación
10. A) verdadero



2

Estudio Preliminar y Gestión del Proyecto

Competencias

Aplica técnicas de recolección de datos, para determinar las necesidades.

Analiza la factibilidad del proyecto de sistemas (operativa, técnica y económica).

Establece los roles del grupo del proyecto.

Organiza las actividades del proyecto.

Cuando haya leído esta unidad los **Resultados de aprendizaje**:

- Identificar técnicas adecuadas para la recolección de datos.
- Identificar las necesidades del cliente para determinar la factibilidad del proyecto de sistemas (operativa, técnica y económica).
- Establecer los roles de los integrantes del grupo del proyecto.
- Organizar las actividades del proyecto utilizando una herramienta para planificación.

Contenidos

2.1 Gestión de Proyectos de Software.

2.2 Problemas y Oportunidades.

2.3 Estudio de Factibilidad.
2.4 Factibilidad Económica.

Introducción al Capítulo

El capítulo aborda la gestión de proyectos, lo que constituye parte esencial de la Ingeniería de software y de todos los desarrollos profesionales de proyectos.

La gestión de proyectos nos ayuda a reconocer los desafíos que nos presenta el cliente al momento de desarrollar un proyecto de software desde su comienzo hasta la culminación del mismo.

La gestión de proyectos permite tener un mejor control de los recursos con los que se cuenta para el desarrollo del proyecto, esto ayuda a poder alcanzar los objetivos planteados por la organización.

En este capítulo hablaremos sobre el estudio preliminar, gestión de proyectos, planificación de proyectos, calendarización, plan de proyectos, etc. En si los temas más relevantes que abarca la gestión de un proyecto de software.

1 Estudio Preliminar y Gestión de Proyecto de Software

La identificación de los requerimientos de software se da a partir de la necesidad de satisfacer el negocio. Para identificar estas necesidades es necesario realizar un estudio preliminar que nos permita determinar e identificar que requiere la empresa como mejora para su desarrollo.

"En el estudio preliminar se definen los límites del sistema y la interacción en el resto de la organización, analizando los diferentes actores y definiendo un modo general el sistema a desarrollar, indicando los tratamientos a utilizar y los flujos de información que se producen" [13]

Toda actividad que permita asegurar la aplicación del proyecto de software, para su eficacia, efectividad y su adaptación al ámbito de la organización, se debe de realizar un estudio preliminar.

Esto se realiza previo a los trabajos que se desarrollan sobre las necesidades que presente el proyecto, ya que este ha de ser expuesto a adaptaciones y modificaciones que requiere por parte del cliente u organización. Un correcto análisis previo al desarrollo del proyecto anticipará los posibles problemas que se puedan presentar y así poder reducir la necesidad de llevar a cabo cambios no previstos en el futuro.

En definitiva, se trata de hacer un estudio previo de la información que aborda el problema que se quiere solucionar, es decir, una detención previa de las necesidades formativas del proyecto.

Una de las prioridades que debe tener el proyecto de software es el establecer el estudio preliminar, ya que esto nos permite

entender mejor la correlación entre la efectividad de la información y el diagnostico de las necesidades del proyecto.

Un buen análisis basado en el diagnóstico y priorización de necesidades tendría un impacto positivo sobre la efectividad del proyecto y contaría con el apoyo que necesita para su implementación.

"Este estudio es posterior a la identificación de las áreas o problemas o necesidades que la empresa tenga. Con él se determinará si la automatización por computador es una solución y si esa solución es viable. Se recomienda que el estudio sea realizado por personas lo más cercanas al problema que se desea solucionar, ya que estos tienen conocimientos de los detalles que deben ser tratados." [14]

Contar desde el inicio del desarrollo del proyecto, con la participación de las personas involucradas ayudará a comprender mejor las necesidades del producto a desarrollar. Se debe trabajar con las personas más cercanas al problema de la organización, ya que estas cuentan con el mayor conocimiento de los detalles que deben ser solucionados.

Un buen estudio preliminar ayudará a tener una mejor automatización del proceso formativo, ya que esto permitirá ajustar los contenidos de la información a los elementos presentes en el proyecto y poder aprovechar estos elementos ya existentes a la gestión de la organización.

El estudio preliminar es un elemento de gran importancia para el éxito de un proyecto de software. Mientras más información esté disponible desde el inicio del desarrollo del proyecto, más precisa será la definición de las metodologías o técnicas a implementar y será más completo para la ejecución el proyecto.

En general, el estudio preliminar nos ayuda a recopilar y analizar la información sobre el proyecto a desarrollar, con lo cual se podrá definir si es factible implementar la automatización por computadora del problema que se requiere solucionar.

1.1 Gestión de Proyectos de Software

Desarrollar un software implica la solución de problemas. En este proceso se incluyen las fases de desarrollo del mismo, y los ciclos de vida del desarrollo del software. El objetivo de disponer de un desarrollo de software es controlar y simplificar los procesos que generan mayor eficiencia y eficacia en el proceso productivo. Por lo tanto se debe llevar el control de todas las actividades que se requieren, esto lo podemos realizar a través de una buena gestión de proyecto de software.

La gestión de proyectos es la capacidad de identificar los desafíos que proporciona el cliente o la organización, y así lograr encontrar, y evaluar las múltiples posibles soluciones que tiene el problema, seleccionando la que más se adapta a las necesidades del proyecto, consiguiendo eficacia y calidad para luego realizarlo.
En la ingeniería de software es esencial contar con una buena gestión de proyectos. Por lo que una mala gestión puede llevar al fracaso del proyecto. La gestión de proyecto ayuda a realizar el seguimiento de los trabajos realizados durante el proceso de desarrollo del software.
Los resultados de este permitirán aprovechar el valor de la tecnología que se ha convertido en un motor de investigación y crecimiento y que de cierta forma ayuda a la automatización del proyecto de software.
Se trata de un proceso continuo, el cual requiere de toda las herramientas necesarias para poderla desarrollar de la mejor manera posible, es una tarea muy importante, ya que esta comienza con un conjunto de actividades que se inician en la fase de planificación del proyecto y continúa durante la fase de desarrollo del mismo.
El ciclo de vida involucrado en la gestión de proyectos, se conforma de dos actividades a realizar: actividades de gestión y las actividades de desarrollo del sistema.
Las actividades de gestión están enfocadas a la administración de las organizaciones, personas, sistemas y procedimientos comprendidos en el proceso de planificación y construcción del sistema.
Las tareas involucradas en el desarrollo del sistema se orientan al desarrollo mismo. Las metodologías de desarrollo están constituidas en diferentes etapas, agrupadas por áreas funcionales de estudio, diseño y construcción.

1.1.1 Actividades de Gestión

"El trabajo difiere enormemente dependiendo de la organización y del producto de software a desarrollar. Sin embargo, en algún momento, muchos gestores son responsables de algunas o de la totalidad de las siguientes actividades:

- Redacción de la propuesta
- Planificación y calendarización del proyecto
- Estimación de costos del proyecto
- Supervisión y revisión del proyecto
- Selección y evaluación del personal
- Redacción y presentación de informes." [15]

La gestión de proyectos de software comprende una gran cantidad de actividades, que contienen la planificación del proyecto, incluyendo el alcance del mismo, estimando costes respecto a la asignación de tareas y gestión de recursos durante el desarrollo del proyecto del software.

La ingeniería de software incluye actividades que van desde la recolección de los requisitos del proyecto, análisis, gestión de procesos, calidad y riesgos, hasta la puesta en marcha del proyecto de software. Dentro de las cuales la más relevante es la de gestionar el proyecto de software.

Las actividades de la gestión de proyectos nos ayudan a cumplir con los objetivos definidos en un tiempo establecido, utilizando recursos tanto humanos como técnicos.

La principales actividades son la de planificar, coordinar, seguimiento y controlar todas las actividades que permitan el alcance del proyecto. Una de las principales etapas es la de redactar la propuesta del proyecto ya que esto permite describir los objetivos del mismo y como se llevaría a cabo para lograr cumplirlos. Lo que incluye la estimación de costes, tiempo y alcance del proyecto.

La planificación del proyecto es una actividad que se realiza antes de la producción del software. Determinando la hitos, entregas y orientación general del responsable del proyecto para conseguir los objetivos establecidos por la organización.

La supervisión del proyecto de software es el seguimiento que se hace del mismo, el cual debe ser continuo. Comparando el progreso de las actividades realizadas para el desarrollo del proyecto.

La revisión de un proyecto se la realiza para verificar que se esté avanzando con el progreso de las actividades. Un proyecto de software puede tener varias revisiones en su gestión. Esta es algo muy importante, ya que de esta revisión dependerá si el proyecto es cancelado.

1.1.2 Planificación del Proyecto

"La planificación es la segunda fase del proceso de gestión de un proyecto, es un preludio necesario a la acción. La planificación generalmente comienza con el objetivo. En la fase de planificación del proyecto, cada tarea y sub-tarea recibe un tiempo razonable para su realización." [16]

Para que un proyecto de software tenga éxito, es necesario tener una buena planificación, la cual debe ser desarrollada con cuidado.

En esta fase se define el alcance del proyecto de software, por tanto la planificación debe ser clara y precisa. Se realiza la planificación de todas las actividades necesarias para llevar a cabo el desarrollo del proyecto de software. Determinar los recursos, tiempo y alcance esperados para la ejecución de cada una de las tareas y funciones propuestas.

"Esta fase debe, en primer lugar, revisar y, sobre todo, obtener un acuerdo o contrato explícito acerca de los temas del proyecto. Esto se obtiene mediante documento de detalle como son las especificaciones del proyecto o lista de trabajos o tareas que hay que realizar." [17]

La identificación de actividades, hitos y entregas del proyecto de software, y también las posibilidades de mitigación de riesgos, se la realiza en la fase de planificación.

Es una disciplina que nos determinar el tiempo en que serán llevadas a cabo las actividades y tareas del proyecto de software.

"El objetivo de la planificación del proyecto de software es proporcionar un marco de trabajo que permita al gestor hacer estimaciones razonables de recursos, coste y planificación temporal. Estas estimaciones se hacen dentro de un marco de tiempo limitado al comienzo de un proyecto de software, y deberían actualizarse regularmente a medida que progresa el proyecto." [18]

En esta fase se establece los objetivos que se desea alcanzar con el desarrollo del proyecto de software. Se identifican los entregables, se planifica el calendario y se elabora planes de contingencia. Los cuales son de apoyo en caso que se presente algún contratiempo en el desarrollo del proyecto.

Se puede realizar una estimación de todas las actividades que implican, planificar a un futuro los inconvenientes y beneficios que se pueden llegar a prever en el desarrollo del software.

1.1.3 El Plan del Proyecto

"El plan de proyecto es un documento formal aprobado y que define cómo debe ser ejecutado, monitoreado y controlado el proyecto. Debe resumir los planes subsidiarios, es decir, los planes específicos de manejo del tiempo, del presupuesto y de la calidad del producto." [19]

El plan de proyecto engloba la información relativa a la gestión del proyecto, es decir cuando llegamos a esta fase, nuestro documento definirá y logrará comunicar el ámbito y recursos al personal de desarrollo y a los clientes. Definiremos y sugeriremos

técnicas de control de riesgos, también definiremos los costes y planificación para la revisión de la gestión.
Como parte principal, les proporcionaremos un enfoque general, para todo el personal relacionado a este proyecto. Y por último describiremos como se garantizará la calidad y se gestionarán los cambios en el mismo.
"El plan de proyecto fija los recursos disponibles, divide el trabajo y crea un calendario de trabajo. En algunas organizaciones, el plan de proyecto es un único documento que incluye todos los diferentes tipos de, planes." [15]
En conclusión, el plan de proyecto es un documento clave a la hora de organizar el trabajo en un desarrollo de software. Porque básicamente plasma sobre el papel todos los recursos disponibles para realizar el desarrollo, y asigna tareas específicas, con fechas precisas y mecanismos concretos de control y seguimiento.

1.1.4 Hitos y Entregas

"Cuando se planifica un proyecto, se debe establecer una serie de hitos que son puntos finales de una actividad del proceso del software. Una entrega es el resultado del proyecto que se entrega al cliente. De forma general, se entrega al final de una fase principal del proyecto como la especificación, el diseño, etc. Como regla general, las entregas son hitos, pero éstos no son necesariamente entregas." [15]
Los Hitos permiten planificar un proyecto, administrar hitos y gestionar los elementos entregables de un proyecto, ya que esta le permite establecer las fechas clave para cada fase del proyecto, así como asociar hitos a listas de tareas.
Cuando hay un retraso en un hito, este afecta a todas las tareas asociadas a él.
Una vez que el hito se marque como terminado, la fecha de este dejará de afectar a las tareas asociadas a él. Así, el estatus de las tareas pendientes quedará intacto.
En las entregas podemos encontrar la visión y alcance y sus definiciones. Es muy importante que el documento de visión y alcances no sea una imposición, debemos esforzarnos para que la visión la compartan todos, la definición del problema, una revisión del proceso existente, una definición general de los requerimientos de usuarios, como está estructurado el proyecto, descripción de cada uno de los roles y una lista de los miembros que lo conforman, las estructuras y los procesos estándar que debe seguir el equipo.

El análisis de riesgos comprende una serie de pasos que hacen que el proyecto se desarrolle de una manera segura, como primer paso, podríamos comenzar con una lluvia de ideas y después de esto, distinguir un riesgo de un problema), identificación y evaluación primaria de riesgos, y los planes para eliminar los riesgos identificados.

1.1.5 Calendarización del Proyecto

"La calendarización del proyecto implica separar todo el trabajo de un proyecto en actividades complementarias y considerar el tiempo requerido para completar dichas actividades. Por lo general, algunas de éstas se llevan a cabo en paralelo." [15]

En esta etapa se define el tiempo de las actividades que implica el desarrollo del proyecto de software. Los gestores de proyecto hacen estimaciones del tiempo y de los recursos que son necesarios para llevar a cabo la realización de las actividades de manera organizada.

Implica separar todo el trabajo del proyecto de software en actividades y considerar el tiempo requerido para completar dichas actividades. Muchas de estas actividades son realizadas en paralelo con otras actividades.

1.2 Problemas y Oportunidades

"Las oportunidades son situaciones que el analista considera susceptibles de mejorar utilizando sistemas de información computarizados. El aprovechamiento de las oportunidades podría permitir a la empresa obtener una ventaja competitiva o establecer un estándar para la industria." [20]

El analista debe observar objetivamente lo que sucede durante el proceso de desarrollo del proyecto de software. Para así, identificar las oportunidades y problemas que éste presente.

Se deben identificar los problemas importantes, aun aquellos que parecen imposibles resolver. El análisis de cada uno de estos problemas se puede determinar si realmente tienen una solución. Lo más importante es continuar indagando y analizando todos los factores hasta poder llegar al núcleo del problema que presenta el proyecto.

"La identificación de objetivos también es una parte importante. En primer lugar, el analista debe averiguar lo que la empresa trata de conseguir. A continuación, podrá determinar si algunas funciones de las aplicaciones de los sistemas de información pueden contribuir a que el negocio alcance sus objetivos aplicándolas a problemas u oportunidades específicos." [20]

El analista se ocupa de identificar los problemas y oportunidades del proyecto de software, esta tapa es muy importante ya que de esto dependerá el éxito del proyecto.
Los problemas se dan a notar de diferentes formas, éstos pueden estar presentes en la organización. Depende en gran parte del analista poder identificar estos problemas y poder analizarlos para solucionarlos. Se deben transformar los problemas en oportunidades.

1.2.1 Analizar las áreas de empresa

"El diagnóstico de la empresa es la consecuencia del análisis de todos los datos relevantes de la misma e informa de sus puntos fuertes y débiles. El diagnostico día a día de la empresa es un herramienta clave, aunque no la única, para la gestión correcta de la empresa. El diagnóstico de la empresa ayuda a conseguir los que podrían considerarse, los objetivos de la mayor parte de las empresas." [21]
El diagnostico debe ser elaborado de forma continua, para así estar pendientes de lo que pasa en el proceso del proyecto de software. Esto también nos permite conocer en cada momento el estado en que se entra la empresa.
Las áreas de la empresa deben ser analizadas, controladas y estar en continuo seguimiento, buscando así la simplificación y eficacia de todos los procesos y recursos de la misma.
Entre las áreas de una organización tenemos lo que es: dirección, administración, planificación, diseño y realización del producto, comercialización, etc.
Se debe identificar los requerimientos de cada área para poder detectar las percepciones de valor que tiene cada área en el proceso de desarrollo del proyecto de software.
También se puede observar, y detectar los recursos empleados en cada área para así poder mejorarlas. Tras un análisis de estas áreas podremos aportar soluciones a los problemas detectados en el proyecto y poder poner en marcha soluciones propuestas.

1.2.2 Técnicas de recolección de datos

La recopilación de datos es una etapa que se debe realizar adecuadamente ya que de esta depende en gran parte como se desarrolle el proyecto de acuerdo a los que los usuarios requieren. Esto nos ayuda a definir adecuadamente las necesidades a las que el proyecto de software se debe adaptar para que este sea de ayuda para la organización.
Para la recolección de datos existen una gran cantidad de técnicas y formas de recolección de datos. Depende del analista

de requerimientos identificar que técnica es la más adecuada para la recolección de información. También debe saber quién le puede proporcionar la información más adecuada posible para el desarrollo del proyecto de software

En la recopilación de datos tenemos distintas formas para recolectar información. Entre las cuales tenemos: entrevistas, cuestionarios, observación, diagramas, casos de uso, etc. Las cuales son de gran ayuda al momento de recolectar información sobre el sistema en el que se está trabajando.

Estas técnicas deben aplicarse adecuadamente dependiendo de la información que se desee obtener para la realización de proyecto de software.

1.2.3 Matriz de Problemas y Oportunidades

"La utilización de esta matriz proporciona al emprendedor una visión global de los principales aspectos positivos y negativos a los que debe enfrentarse su idea de negocio, ayudándolo de forma intuitiva a valorar si vale la pena seguir adelante con dicha idea." [22]

La matriz de problemas y oportunidades también conocido como matriz FODA es una matriz que nos permite identificar las oportunidades y problemas que se presenten el desarrollo del software.

La matriz FODA es un instrumento que sirve para realizar un análisis organizacional de un empresa, determinando los factores que determinan el éxito o fracaso.

La matriz de problemas y oportunidades nos permite conocer las fortalezas, debilidades, amenazas y oportunidades que se nos pueden presentar en la realización de un proyecto.

Las fortalezas son las capacidades con que la empresa y que nos permite llegar al éxito. Las debilidades son los aspectos vulnerables de la empresa. Estos pueden ser carencia de recursos, habilidades que no se poseen, y actividades que nos e desarrollan a tiempo, etc.

Las amenazas son las situaciones del entorno que pueden tener aspectos negativos que pueden representar peligros en la organización de la empresa. Las oportunidades son los aspectos positivos, favorables del entorno que permite obtener ventaja competitiva.

Mediante la matriz FODA se puede determinar cuál es la situación del objeto de estudio ya que proporciona información detallada sobre los factores internos y externos que intervienen en este

La matriz FODA permite buscar y analizar, de forma proactiva y sistemática, todas las variables que intervienen en el proyecto con el fin de tener más y mejor información al momento de tomar decisiones en el proyecto

1.2.4 Ámbito del Software

"El estudio del ámbito del software, consiste en realizar un estudio del ámbito al cual pertenecen los componentes software para los que se quiere evaluar la calidad. Es recomendable revisar fuentes de información que describan dicho ámbito. " [23]

Una de las actividades del proyecto de software que se debe realizar es el ámbito del software en donde se describirá el control y los datos a procesar, la función, el rendimiento, las restricciones, las interfaces y la fiabilidad. Se evalúan las funciones descritas en la declaración del ámbito, y en algunos casos se refinan para dar más detalles antes del comienzo de la estimación.

Además se deben definir los escenarios de peor caso y mejor caso de manera que los resultados del proyecto puedan limitarse.

El cliente es el único que puede ayudarnos a determinar el ámbito, por lo tanto la comunicación con el cliente es elemental, es por esto que se emplea una técnica muy común que es una reunión o una entrevista.

El ámbito de software implica la determinación de lo que se va a emplear para llevar a cabo el desarrollo de software. Es considerada como la primera actividad que se lleva a cabo en la administración de proyectos de software, en ello incluye la obtención de información necesaria para ello el analista se reúne con el cliente para determinar puntos de interés que se incluirán en el software.

La recolección de datos es importante ya que esta permite acercarse al cliente para establecer una comunicación y así poder obtener la información necesaria para establecer los requerimientos del software.

EL ámbito del software comprende la estimación de recursos, duración, esfuerzo y costos. Cuando se planifica un proyecto se tiene que obtener las estimaciones de los recursos requeridos para el desarrollo del software.

Estos recursos pueden ser recursos humanos, que son aquellas personas que intervienen en la planificación y desarrollo del software.

También recursos de hardware y software, se debe considerar las distintas herramientas que ayudan al desarrollo del software.

1.3 Estudio de Factibilidad

"Es necesario determinar si los proyectos seleccionados son factibles. La definición de factibilidad va mucho más allá del uso común del término. Para los proyectos de sistemas, la factibilidad es valorada en tres formas principales: operacional, técnica y económicamente. Un proyecto debe ser factible en las tres formas para merecer un desarrollo posterior." [24]

El estudio de factibilidad es el proyecto que se basa en presentar la información necesaria a la institución en el cual se ven reflejados los datos importantes para saber si el proyecto es útil para la institución, tomando en cuenta los recursos disponibles que la institución puede proporcionar, los estudios de factibilidad nunca deben incluirse recursos que la institución no es capaz de proveer.

Para tomar la decisión de involucrarse en un proceso de integración se deberá de realizar un análisis lo más completo posible para determinar si es conveniente invertir, aunque se tengan que enfrentar diversos riesgos.

Al tomar una decisión de cambio en la empresa se debe realizar este tipo de estudio; ya que esta investigación puede ayudar a evitar que un proyecto fracase, ayudando a las personas a conocer y disminuir los riesgos, y asegurar el éxito del estudio de la empresa.

Mediante el estudio de factibilidad se puede determinar cuáles serán los diferentes factores que intervendrán en el desarrollo del proyecto, y de esta manera analizar si todos los componentes están en condiciones favoriticas para que se ejecute logrando así tomar decisiones asertivas que benefician al desarrollador.

1.3.1 Factibilidad Operativa

"Requiere creatividad por parte del analista de sistemas, así como de su capacidad de persuasión para indicarle a los usuarios cuáles son las probables interfaces y cuáles satisfarán sus necesidades. El analista de sistemas también debe escuchar con atención lo que realmente quieren los usuarios y lo que al parecer utilizarán." [24]

La factibilidad operativa identifica si el proyecto puede ser operado a través de los recursos de la organización, además de los recursos que participaran en el proyecto. Busca la manera de tener la mejor disponibilidad del momento y lugar adecuado, cuando el proyecto se convierta en resultados.

1.3.2 Factibilidad Técnica

"Una gran parte de la determinación de recursos tienen que ver con la valoración de la factibilidad técnica. El analista debe encontrar si los recursos técnicos actuales pueden ser mejorados o añadidos, en forma tal que satisfagan la petición bajo consideración." [15]

La factibilidad técnica hace referencia a la posibilidad que tiene el proyecto de satisfacer los requerimientos del equipamiento tecnológico con el que cuenta la organización.

En esta fase se debe identificar si los recursos tecnológicos tanto de software como hardware, con el que cuenta la organización, son los suficientes para la puesta en marcha del proyecto, si hay que disponer de nuevos equipos tecnológicos.

1.3.3 Factibilidad Económica

"La empresa interesada debe tener la capacidad de calcular el valor de la inversión bajo evaluación antes de comprometerse a un estudio de sistemas completo. Si los costos a corto plazo no son opacados por las ganancias a largo plazo o no producen una reducción inmediata de los costos operativos, el sistema no es económicamente viable y el proyecto debe detenerse." [24]

1.4 Análisis de la Factibilidad Económica

Este tipo de factibilidad se ocupa de identificar con la mayor precisión posible del coste que implica el instalar y desarrollar el proyecto de software.

Aborda todas las actividades que nos ayudan a identificar los beneficios y costos del proyecto, y a saber si el proyecto es económicamente viable.

La factibilidad económica permite identificar la obtención de las ganancias que se puede obtener con el desarrollo del proyecto de software.

Presenta el alcance que tendrá el proyecto respecto a lo técnico y económico.

1.4.1 Presupuesto

Es un tipo de documento en el cual se tiene una previsión de los gastos, ingresos relativos a las actividades económicas que implican el desarrollo del software. Es considerado un plan financiero.

Se formula para lograr en un cierto periodo de tiempo, si se han alcanzado los objetivos propuestos, los cuales son expresados en términos monetarios.

Hacer un presupuesto es planear que es lo que se quiere logar en el futuro y poder expresarlo en manera económica

Es una manera de tener una organización de los gastos que implica el desarrollo del proyecto.

1.4.2 Matriz de Costo - Beneficio

El análisis del costo-beneficio se lo realiza a lo largo de la vida del proyecto de software

Es una herramienta que permite oponer en equilibrio los costos y beneficios del proyecto. Se debe examinar las ganancias que se revisen con el desarrollo del producto.

A partir de esto podemos estimar el impacto financiero acumulado de lo que queremos lograr con el proyecto.

Nos ayuda a la toma de decisiones, ya que hace una comparación entre los costos y beneficios de las diferentes actividades realizadas.

1.4.3 Amortización y Rentabilidad

Las amortizaciones no son pagos, por lo tanto no se tienen en cuenta directamente al evaluar la rentabilidad de un proyecto. La amortización es una notación en los libros contables de la depreciación de un activo en un determinado periodo.

La evaluación de un proyecto debe estar basada en los flujos de efectivo del proyecto. Es una relación que existe entra la utilidad y la inversión necesaria para llevar a cabo el proyecto.

CONCEPTOS IMPORTANTES

- **Actividades de gestión**: son todas las actividades que implican el desarrollo del proyecto de software. Cumpliendo los objetivos establecidos por la organización.
- **Calendarización de proyectos**: esta fase nos permite tener un mejor control de tiempo de las actividades a realizar, para que así no halla contratiempos en el desarrollo del proyecto
- **Estudio de factibilidad:** es un análisis de los aspectos técnicos, económico y operacional de un proyecto. De este análisis depende si el proyecto tendrá éxito.
- **Gestión de proyectos**: es una disciplina que implica tener una buena organización, motivación y control de todos los recursos con el propósito de alcanzar los objetivos propuestos.
- **Planificación de proyectos:** es una etapa de gran importancia donde se identifica como se debe de llevar a cabo un proyecto en un plazo determinado.

LECTURAS COMPLEMETARIAS

Del artículo: "**Gestionando Pequeños Proyectos"** de Norberto Figuerola, disponible en: https://articulospm.files.wordpress.com/2015/06/gestionando-pequec3b1os-proyectos.pdf

Desarrolle las siguientes preguntas

1. ¿Cómo se definen las mejores prácticas en la gestión de proyecto?
2. ¿A qué denominamos Body of Knowledge?
3. Mencione una práctica para la gestión de proyectos
4. ¿Cómo definiría al PMBOK?
5. Realice una valoración crítica sobre el artículo

Del artículo: "**Determinación de riesgos en entornos muy interactivos. ¿Cómo evitar el factor Titanic en su proyecto?"** de Weeks Bruce, disponible en: http://americalatina.pmi.org/latam/KnowledgeCenter/Articles/~/media/726EB975FA494F2EA7BC96AD09D9398D.ashx

Desarrolle las siguientes preguntas

1. ¿Cómo definimos el hundimiento del Titanic?
2. ¿Qué son los compartimentos estancos, cómo podría relacionarlo con un proyecto de software?
3. ¿Qué piensa sobre las órdenes del Capitán justo antes dela colisión?
4. ¿Cómo se deberían determinar las interacciones entre riesgos?
5. De su conclusión sobre la pregunta del artículo ¿Cuánta gestión de riesgos es suficiente?

TALLER

Taller Nº 1 – Gestión de Proyectos

1. **¿Cuál es el objetivo del desarrollo de software?**

   _____

   _____

   _____

2. **¿Qué es la Gestión de Proyectos?**

3. **¿Cuáles son las actividades de la Gestión de Proyectos?**

4. **Escriba la definición de la planificación del proyecto**

5. **¿A qué se refiere el Plan del Proyecto?**

Taller Nº 2 – Problemas y Oportunidades

1. **¿A Qué se refieren las oportunidades en el proyecto?**

2. **¿Cuáles son las actividades que realiza el analista de requerimientos**?

3. **¿Qué se determina con la matriz FODA?**

4. **¿En qué consiste el ámbito del Software?**

_____
_____
_____

5. ¿Por qué es importante analizar las áreas de la empresa?

_____
_____
_____

Taller Nº 3 – Estudio de Factibilidad

1. **Realice un supuesto estudio de factibilidad técnica para un sistema de facturación de una empresa de confección de ropa deportiva en la ciudad de Machala.**

2. **De acuerdo al texto analice y realice un estudio de factibilidad Operativa:**

> *"La empresa ANUNICO Ltda. Es una empresa dedicada al arrendamiento de residencias para estudiantes. Esta empresa presenta problemas en el control de la información y disponibilidad de residencias debido a la falta de un espacio online con capacidades claras para generar eficiencia en la gestión oferta y demanda de residencias por parte de los estudiantes.*
>
> *Durante ya muchos años la empresa ANUNICO Ltda. Ha venido trabajando bajo una mala gestión de la información de los estudiantes, también presenta una mala organización en lo concerniente a la disponibilidad de residencias. Esta mala gestión de la información ha provocado además que el esfuerzo por parte de los usuarios sea excesivo, debido a que el registro de los estudiantes se realiza de forma manual, trayendo como consecuencia una mala atención y un desconocimiento de las ofertas que se brindan en esta empresa".*

3. **De acuerdo a la tabla de análisis de costo – beneficio que se puede concluir:**

Análisis costo – beneficio

Concepto:	Año 1	Año 2	Año 3	Año 4	Año 5
Costos	18.055.85	21.847.58	26.435.58	31.987.05	38.704.33
Beneficios	5.250.89	6.353.58	7.687.84	9.302.28	11.255.77

EVALUACIÓN

1	**FORMATO**	**SIMPLE**
	Contexto	
	Planteamiento	**Durante la supervisión de un proyecto se puede predecir problemas importantes solo con hablar con el personal involucrado.**
	OPCIÓN A	Verdadero
	OPCIÓN B	Falso
	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios
2	**FORMATO**	SIMPLE
	Contexto	

	Planteamiento	**La calendarización no permite establecer actividades paralelas porque los recursos no se comparten.**
	OPCIÓN A	Verdadero
	OPCIÓN B	Falso
	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios
3	**FORMATO**	SIMPLE
	Contexto	
	Planteamiento	**En esta actividad de gestión del proyecto de software se compara el progreso con los costes actuales y los planificados.**
	OPCIÓN A	Planificación
	OPCIÓN B	Selección y evaluación de personal
	OPCIÓN C	Supervisión
	OPCIÓN D	Redacción de informes
	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios
4	**FORMATO**	SIMPLE
	Contexto	
	Planteamiento	**Son situaciones negativas, externas al proyecto, que pueden atentar contra este.**
	OPCIÓN A	Amenazas
	OPCIÓN B	Debilidades
	OPCIÓN C	Fortalezas
	OPCIÓN D	Oportunidades
	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios
5	**FORMATO**	SIMPLE
	Contexto	

	Planteamiento	**Muestra cómo puede cambiar el sistema el entorno del usuario y las actividades que realiza. Esto lo hace el Estudio de Factibilidad.**
	OPCIÓN A	Jurídica
	OPCIÓN B	Operativa
	OPCIÓN C	Técnica
	OPCIÓN D	Económica
	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios
	FORMATO	SIMPLE
	Contexto	
	Planteamiento	**Quien es el responsable de la planificación y temporalización del desarrollo del proyecto de software.**
	OPCIÓN A	Business Analyst
	OPCIÓN B	Development Manager
	OPCIÓN C	Project Manager
	OPCIÓN D	Software Manager
	Respuesta Correcta	
	Nivel	1
6	**Operación Cognitiva**	Aplicación de Conceptos y Principios
	FORMATO	SIMPLE
	Contexto	
	Planteamiento	**Llegar a la determinación de que los requerimientos originales es consecuencia de la:**
	OPCIÓN A	Redacción de informes
	OPCIÓN B	Supervisión del proyecto
	OPCIÓN C	Planificación del proyecto
	OPCIÓN D	Redacción de la propuesta del proyecto
	Respuesta Correcta	
7	**Nivel**	1

	Operación Cognitiva	Aplicación de Conceptos y Principios
8	**FORMATO**	SIMPLE
	Contexto	
	Planteamiento	**Normalmente todo _____ es un _____**
	OPCIÓN A	Hito-entregable
	OPCIÓN B	Entregable-documento
	OPCIÓN C	Entregable-hito
	OPCIÓN D	Documento-hito
	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios
9	**FORMATO**	SIMPLE
	Contexto	
	Planteamiento	**Además de determinar la viabilidad el proyecto, el Estudio Preliminar también formula recomendaciones para mejorar los procesos analizados.**
	OPCIÓN A	Verdadero
	OPCIÓN B	Falso
	Respuesta Correcta	
	Nivel	**1**
	Operación Cognitiva	**Aplicación de Conceptos y Principios**
10	**FORMATO**	SIMPLE
	Contexto	
	Planteamiento	**Para que un proyecto sea factible basta con establecer que lo beneficios son superiores a los costos.**
	OPCIÓN A	Verdadero
	OPCIÓN B	Falso
	OPCIÓN C	

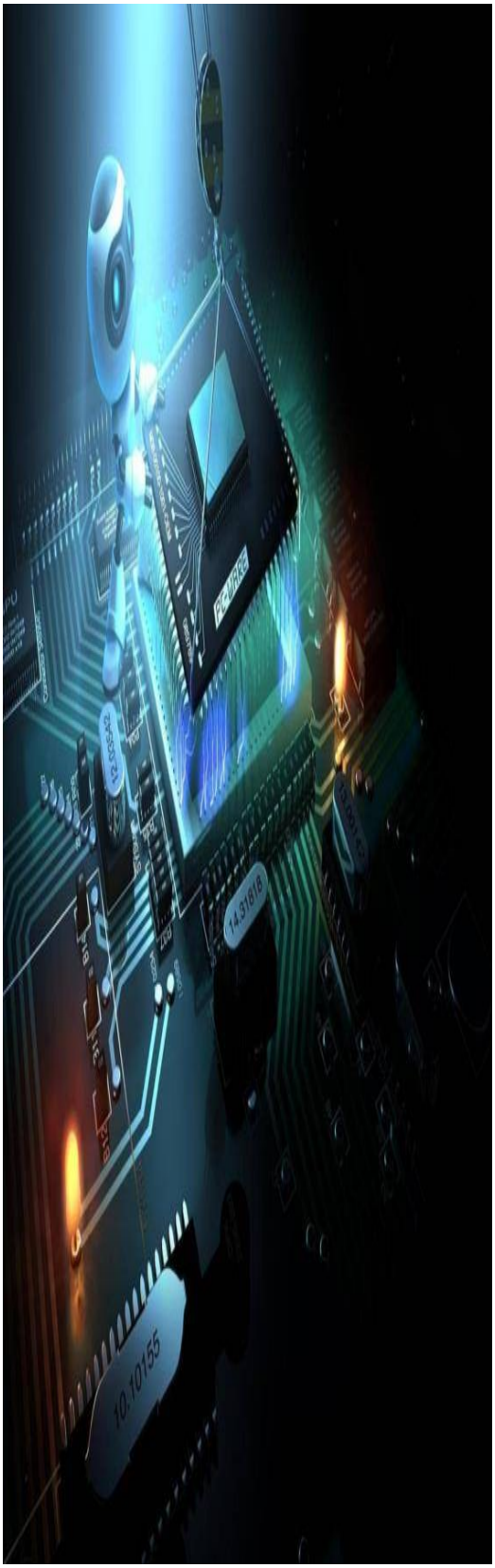
	OPCIÓN D	
	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios

GLOSARIO

- **Administración**: es el proceso de planificar, organizar, dirigir y controlar un proyecto. Para logar alcanzar los objetivos establecidos por la organización.
- **Ámbito**: es el espacio de un sitio o lugar. Es el área comprendida entre límites determinados.
- **Amortización**: término económico, que hace referencia al proceso de distribuir el tiempo de un valor verdadero. Es una representación de la salida de efectivo.
- **Calendarización**: es una etapa en la cual se determina anticipadamente las fechas que deben culminar cierta actividad del proceso de desarrollo del software.
- **Coste**: es el valor monetario invertido por una empresa para lograr la producción de un producto o servicio. Es un gasto que representa la fabricación de un producto.
- **Datos**: son representaciones simbólicas de una variable cuantitativa o cualitativa. Estos pueden contener información.
- **Estimación**: se define al conjunto de técnicas que sirven para tener un valor aproximado de un producto.
- **Factibilidad:** es la disponibilidad de los recursos necesarios para el desarrollo de un proyecto. Llevando a cabo los objetivo y metas propuestos.
- **Requerimientos**: son exigencias que tiene que cumplir el sistema para poder funcionar correctamente. Técnicas y procedimientos que permiten identificar elementos necesarios para definir un proyecto de software.
- **Software**: es conjunto de programas que permiten la interacción entre una máquina y un usuario. Componentes lógicos necesarios para la realización de tareas.

SOLUCIONARIO (EVALUACIÓN)

1. A) Verdadero
2. B) Falso
3. C) Supervisión
4. A) Amenazas
5. B) Operativa
6. C) Project Manager
7. B) supervisión del proyecto
8. C) Entregable – hito
9. A) Verdadero
10. B) Falso



3

Ingeniería de Requerimientos

Competencias

Establece correctamente los requisitos funcionales y no funcionales del proyecto.

Identifica la estrategia adecuada para la obtención de requerimientos.

Cuando haya leído esta unidad los **resultados de aprendizaje**:

- Identificar correctamente los requisitos funcionales y no funcionales del proyecto.
- Aplicar estrategias adecuadas para la obtención de requisitos.

Contenidos

3.1 Especificación de Requisitos.
3.2 Estándar IEEE

Introducción al Capítulo III

La fase en la que se identifican y comprenden los requerimientos necesarios para llevar a cabo un sistema suele ser la más difícil en algunos casos pero también la más importante. Pues estos requerimientos están enfocados a las necesidades tanto del cliente como del usuario lo cual permite al desarrollador tener una visión clara de lo que va a construir.

Los requerimientos de un sistema vienen a ser las especificaciones de los servicios otorgados por la aplicación, muchas de las veces el cliente no tiene claro lo que desea realizar, además los usuarios que manejan el sistema no siempre identifican claramente las nuevas características que podrían ser de beneficio al sistema, debido a ello el desarrollador debe saber cómo llegar a los requerimientos necesarios que le serán de ayuda en el desarrollo del software.

A este proceso de identificar los requerimientos más su análisis, documentación y verificación se le conoce como Ingeniería de requerimientos. Este capítulo engloba los temas más sobresalientes que se puedan considerar en la Ingeniería de requerimientos, en si menciona su división tanto en requerimientos funcionales y no funcionales así como la organización de estos en un documento de especificación de requerimientos.

3 Ingeniería de Requerimientos

"La Ingeniería de Requerimientos es la disciplina para desarrollar una especificación completa, consistente y no ambigua, la cual servirá como base para acuerdos comunes entre todas las partes involucradas y en dónde se describen las funciones que realizará el sistema. En este workshop el instructor presentará los principios y técnicas necesarios de la Ingeniería de Requerimientos como pieza inicial de un proyecto exitoso que termine en un "contrato" que se pueda cumplir en un tiempo, presupuesto y alcance inicialmente establecidos, utilizando estándares mundialmente conocidos y aceptados por empresas líderes en el desarrollo de software (IEEE, CMMI, RUP)" [25]

3.1 Especificación de Requisitos

"La especificación de requisitos del software se produce como culminación de la tarea de análisis. En ella se establece una descripción completa de la información, una descripción funcional, una indicación de los requisitos de rendimiento y de las restricciones de diseño y unos criterios de validación apropiados." [26]

"Los requerimientos para un sistema son la descripción de los servicios proporcionados por el sistema y sus restricciones operativas. Estos requerimientos reflejan las necesidades de los clientes de un sistema que ayude a resolver algún problema como el control de un dispositivo." [27]

"Algunos de los problemas que surgen durante el proceso de ingeniería de requerimientos son resultado de no hacer una clara separación entre estos diferentes niveles de descripción. Aquí se distinguen utilización la denominación requerimientos del usuario para designar los requerimientos abstractos de alto nivel, y requerimientos de los sistemas para designar la descripción detallada de los que el sistema debe hacer." [27]

La especificación de requerimientos presenta una descripción detallada de la información requerida para llevar a cabo el software, en sí esta fase que se da después del estudio de factibilidad es necesaria y se podría decir que la más importante, pues aquí se encuentra de una manera específica el funcionamiento del sistema y la manera en que los usuarios que interactuaran con él.

3.1.1 Requerimientos Funcionales y No Funcionales

"Los analistas de sistemas pueden clasificar los requerimientos identificados en dos grandes grupos: los requerimientos funcionales y los requerimientos no funcionales." [28]

"A menudo, los requerimientos de sistemas software se clasifican en funcionales y no funcionales, o como requerimientos del dominio:" [27]

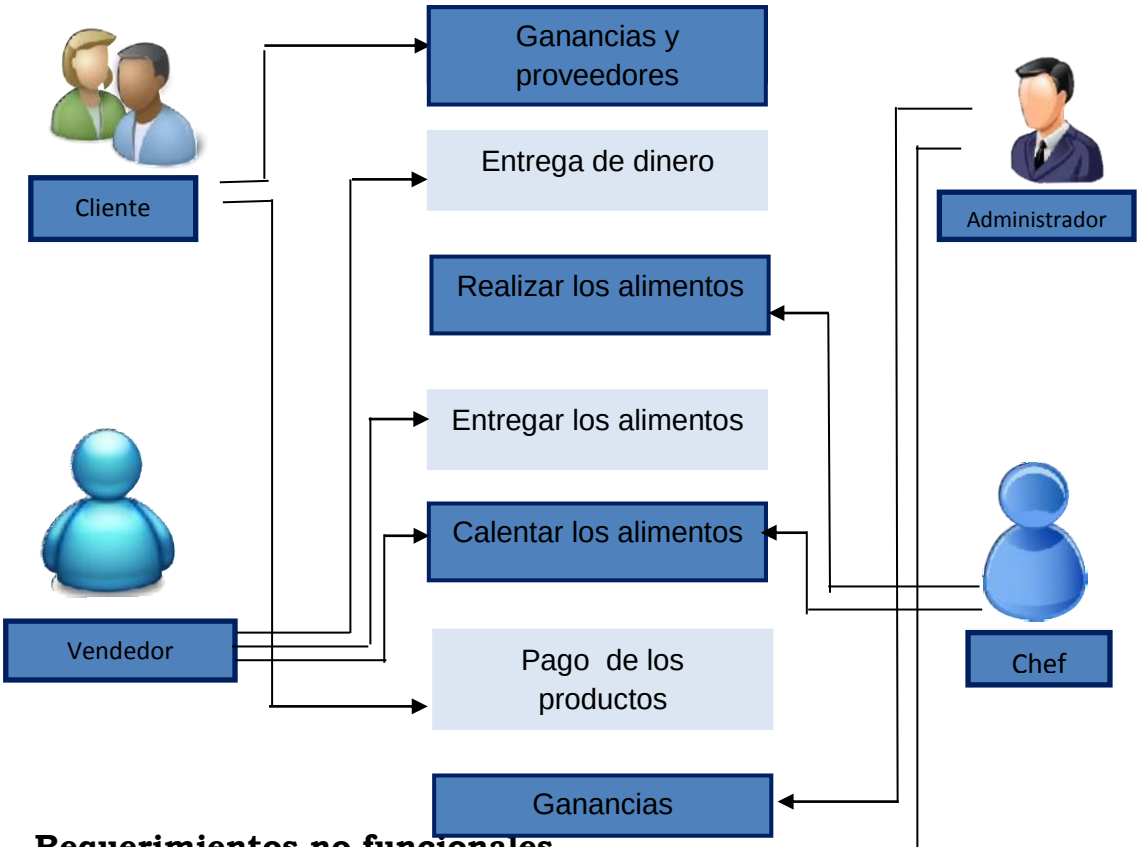
Requerimientos funcionales

"Los requerimientos funcionales hacen referencia a la descripción de las actividades y servicios que un sistema debe proveer. Normalmente este tipo de requerimientos están vinculados con las entradas, las salidas de los procesos y los datos a almacenar en el sistema." [28]

Son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que este debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos, los requerimientos funcionales de

los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer."

Los requerimientos funcionales de un sistema describen lo que el sistema debe hacer. Los cuales dependen del software a desarrollar, los cuales en su mayor parte se los redactan de una forma abstracta. Pero estos describen con detalle la función de este, sus entradas y salidas, excepciones, etcétera.



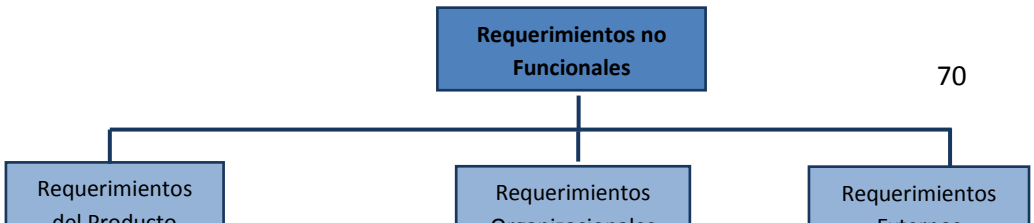
Requerimientos no funcionales

"Por otra parte los requer[illegible]les describen otras prestaciones, características y limitaciones que debe tener el sistema para alcanzar el éxito. Los requerimientos no funcionales engloban características como rendimiento, facilidad de uso,

Ilustración 3: Requerimientos funcionales

[illegible] auditorías internas" [28]

Son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Los requerimientos no funcionales a menudo se aplican al sistema en su totalidad. Normalmente apenas se aplican a características o servicios individuales del sistema." [27]



“Los requerimientos no funcionales, como su nombre sugiere, son aquellos requerimientos que no se refieren directamente a las funciones específicas que proporciona al sistema, sino a las propiedades emergentes de este como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento. De forma alternativa, definen las restricciones del sistema como la capacidad de los dispositivos de entrada/ salida y las representaciones de datos que se utilizan en las interfaces del sistemas.” [27]

“Los requerimientos no funcionales rara vez se asocian con características particulares del sistema. Especifican o restringen las propiedades emergentes del sistema.” [27]

Los requerimientos no funcionales pueden ser tratados como el “cómo hacer”, es decir se basa en los parámetros a considerar para llevar a cabo el sistema o también se podría decir que son las restricciones que este tiene al momento de llevar a cabo una operación.

Tipos de Requerimientos no funcionales

“Requerimientos del producto: Estos requerimientos especifican el comportamiento del producto. Algunos ejemplos son los requerimientos de rendimiento en la rapidez de ejecución del sistema y cuanta memoria se requiere.” [27]

“Requerimientos organizacionales: Estos requerimientos se derivan de políticas y procedimientos existentes en la organización del cliente y en la del desarrollador. Algunos ejemplos son los estándares en los procesos que deben utilizarse, como los lenguajes de programación o el método de diseño a utilizar.” [27]

Los requerimientos organizacionales son los acuerdos a los que el desarrollador y el cliente llegan para aplicarse al software

"Requerimientos externos: Este gran apartado incluye todos los requerimientos que se derivan de los factores externos al sistema y de su proceso de desarrollo, Estos pueden incluir los requerimientos de interoperabilidad que definen la manera en que el sistema interactúa con sus sistemas de otras organizaciones. Los requerimientos legislativos que deben seguirse para asegurar que el sistema funciona dentro de la ley, y los requerimientos éticos." [27]

3.1.2 Requerimientos del Dominio

"Los requerimientos del dominio se derivan del dominio de aplicación del sistema más que de las necesidades específicas de los usuarios. Normalmente incluyen terminología especializada del dominio o referencia a conceptos del dominio. Pueden ser requerimientos funcionales nuevos. Restringir los existentes o establecer como se deben ejecutar cálculos particulares." [27]

Este tipo de requerimiento se basa en las características o fundamentos del dominio del sistema, es importante hacer que estos requerimientos se satisfagan para que el sistema funcione correctamente.

3.1.3 Requerimientos del Usuario

"Los requerimientos del usuario para un sistema deben describir los requerimientos funcionales y no funcionales de tal forma que sean compresibles por los usuarios del sistema sin conocimiento técnico detallado." [27]

"Sin embargo, pueden surgir diversos problemas cuando se redactan con frases del lenguaje natural en un documento de texto." [27]

- "**Falta claridad:** Algunas veces es difícil utilizar el lenguaje de forma precisa y no ambigua sin hacer el documento poco conciso y difícil de leer.
- **Confusión de requerimientos:** No se distinguen claramente los requerimientos funcionales y no funcionales, las metas del sistema y la información para el diseño.
- **Conjunción de requerimientos:** Diversos requerimientos diferentes se pueden expresar de forma conjunta como un único requerimiento." [27]

Los requerimientos de usuario son indispensables al momento de desarrollar el software ya que se centra en sus necesidades, el desarrollador debe comunicarse con el usuario por medio de un

lenguaje natural o diagramas de tal manera que sea fácil de entender, es muy importante considerar la opinión del usuario mientras se vaya avanzando en el proyecto pues es este quien le dará uso y de ello depende que nuestro proyecto tenga éxito.

3.1.4 Requerimientos del Sistema

"Los requerimientos del sistema son versiones extendidas de los requerimientos del usuario que son utilizados por los ingenieros de software como punto de partida para el diseño del sistema. Agregan detalle y explican como el sistema debe proporcionar los requerimientos del usuario." [27]

Los requerimientos del sistema se puede decir que son los que dan inicio a esta fase, pues de estos dependen los requerimientos del usuario y en si es a este a quien se le va a aplicar los cambios.

"En teoría, los requerimientos del sistema simplemente deben describir el comportamiento externo del sistema y sus restricciones operativas. No deben tratar de cómo se debe diseñar o implementar." [27]

"El análisis de requerimientos del sistema ofrece una visión global del proyecto, realizando el estudio de las necesidades operacionales de los usuarios, determinando las restricciones sobre las que tendrá que operar el sistema y definiendo las funciones que debe llevar a cabo." [29]

Los requerimientos del sistema vienen a ser requisitos funcionales pues refleja las operaciones que el sistema llevará a cabo, es decir mediante este tipo de requerimientos se puede analizar el comportamiento del sistema.

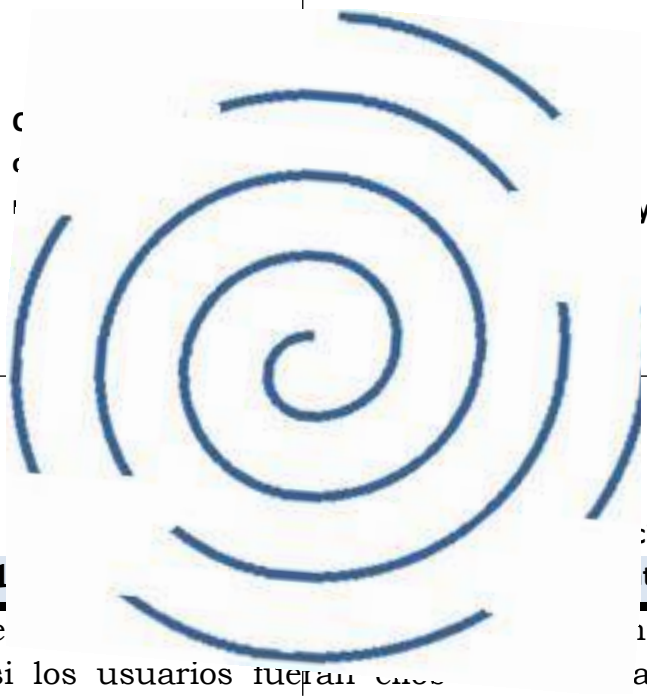
3.1.5 Obtención y análisis de requerimientos

"En este apartado se describirán los requisitos que se desean que tenga el modelo de datos que se va a diseñar, comentando a que se dedica la empresa imaginaria que se ha creado y la actividad que deberá realizar." [30]

"La siguiente etapa del proceso de ingeniería de requerimientos es la obtención y análisis de requerimientos. En esta actividad, los ingenieros de software trabajan con los clientes y los usuarios finales del sistema para determinar el dominio de la aplicación, qué servicios deben proporcionar el sistema, el rendimiento requerido del sistema, las restricciones hardware, etc." [27]

"La obtención y análisis de requerimientos pueden afectar a varias personas de la organización. El termino stakeholder se utiliza para referirse a cualquier persona o grupo que se verá afectado por el sistema, directa o indirectamente. Entre los

stakeholders se encuentran los usuarios finales que interactúan con el sistema y todos aquellos en la organización que se pueden ver afectados por su instalación. Otros stakeholders del sistema pueden ser los ingenieros de desarrollo o dan mantenimiento a otros sistemas relacionados, los gerentes del negocio, los expertos en el dominio del sistema y los representantes de los trabajadores." [27]



3.1.6 Perfil d[illegible]ción de [illegible]tos

"Los de[illegible]nterfaz de usuario como si los usuarios fueran [illegible] a que tienen una cultura profesional y unos conocimientos de informática muy c[illegible]

Ilustración 5: Proceso de obtención y análisis de requerimientos

Los perfiles de usuario tienen estos aspectos:

- Experiencia en torno a hardware y software (ratón, teclado, ventas, etc.)
- Experiencia en aplicaciones del mismo
- Experiencia en el trabajo
- Frecuencia de uso del software y grado de rotación del personal." [31]

El usuario es quien manipula el sistema y este puede cumplir diferentes actividades. El usuario debe estar informado de los procesos que se están llevando a cabo en el sistema pues es este quien realizará las tareas del mismo.

3.2 Estándar IEEE

"Las características deseables para una buena especificación de requisitos software que se indican en el IEEE son las siguientes:

- Correcta
- No ambigua
- Completa
- Verificable
- Consistente
- Clasificada
- Modificable
- Explorable
- Utilizable durante las tareas de mantenimiento y uso" [32]

Corrección

"La Especificación de Requisitos de Software (ERS) es correcta si y sólo si todo requisito que figura en ella refleja alguna necesidad real. La corrección de la ERS implica que el sistema implementado será el sistema deseado." [32]

Ambigüedad

"Un documento es no ambiguo si y solo si cada requisito descrito tiene una única interpretación. Cada característica del producto final debe ser descrita utilizando un término único y, en caso de que se utilicen términos similares en distintos contextos, se deben indicar claramente las diferencias entre ellos. Incluso se puede incluir un glosario en el que indicar cada significado específicamente. Los analistas deben poner un cuidado especial a la hora de especificar los requisitos. El hecho de utilizar el lenguaje natural para hacer la ERS comprensible a los usuarios supone un riesgo muy elevado, porque el lenguaje natural puede llegar a ser muy ambiguo." [32]

Completa

Una ERS es completa si:

- "Incluye todos los requisitos significativos del software (relacionados con la funcionalidad, ejecución, diseño, atributos de calidad o interfaces externas).
- Existe una definición de respuestas a todas las posibles entradas, tanto válidas como inválidas, en todas las posibles situaciones.
- Cumple con el estándar utilizado. Si hay alguna parte del estándar que no se utiliza, se debe razonar suficientemente por qué no se ha utilizado dicho apartado.
- Aparecen etiquetadas todas las figuras, tablas, diagramas, etc., así como definidos todos los términos y unidades de medida empleados." [32]

Verificabilidad

"Un requisito se dice que es verificable si existe algún proceso no excesivamente costoso por el cual una persona o una máquina pueda chequear que el software satisface dicho requerimiento" [32]

Consistencia

"Una ERS es consistente si y sólo si ningún conjunto de requisitos descritos en ella son contradictorios o entran en conflicto. Se pueden dar tres casos:

- Requisitos que describen el mismo objeto real utilizando distintos términos.
- Las características especificadas de objetos reales. Un requisito establece que todas las luces son verdes y otro que son azules.
- Conflicto lógico o temporal entre dos acciones determinadas. Se llega a un punto en el que dos acciones serían perfectamente válidas (¿sumar o multiplicar?)" [32]

Clasificación

"No todos los requisitos son igual de importantes. Los requisitos pueden clasificarse por diversos criterios:

- Importancia: Pueden ser esenciales, condicionales u opcionales.
- Estabilidad: Cambios que pueden afectar al requisito" [32]

Modificabilidad

"Una ERS es modificable si cualquier cambio puede realizarse de manera fácil, completa y consistente. Para ello, es deseable tener una organización coherente y fácil de usar en la que aparezca el índice o una tabla de contenidos fácilmente accesible." [32]

Explorabilidad

"Una ERS es explorable si el origen de cada requerimiento es claro tanto hacia atrás (origen que puede ser un documento, una persona etc.) como hacia delante (componentes del sistema que realizan dicho requisito).

Cuando un requisito de la ERS representa un desglose o una derivación de otro requisito, se debe facilitar tanto las referencias hacia atrás como hacia adelante en el ciclo de vida." [32]

Utilizable durante las tareas de mantenimiento y uso

"En la ERS también se deben tener en cuenta las necesidades de mantenimiento. El personal que no ha

intervenido directamente en el desarrollo debe ser capaz de encargarse de su mantenimiento. Así, dicha ERS actúa a modo de plano de la aplicación, permitiendo incluso modificaciones que no requieran un cambio en el diseño." [32]

3.2.1 Documento de Requerimientos de Software estándar IEEE 830-1998

"El Estándar más ampliamente conocido es el IEEE/ANSI 830-1998 (IEEE, 1998). Este estándar IEEE sugiere la siguiente estructura para los documentos de requerimientos:

1. Introducción
   1.1 Propósito del documento de Requerimientos
   1.2 Alcance del producto
   1.3 Definiciones, acrónicos y abreviaturas
   1.4 Referencias
   1.5 Descripción del resto del documento
2. Descripción general
   2.1 Perspectiva del producto
   2.2 Funciones del producto
   2.3 Características del usuario
   2.4 Restricciones generales
   2.5 Suposiciones y dependencias
3. Requerimientos específicos: incluyen los Requerimientos funcionales, no funcionales y de interfaz. Obviamente, esta es la parte más sustancial del documento, pero debido a la alta variabilidad en la práctica organizacional, no es apropiado definir una estructura estándar para esta sección." [27]

CONCEPTOS IMPORTANTES

- **Documento de requerimientos:** es el resultado detallado de los requisitos obtenidos en la fase de análisis que sirve para llevar a cabo el sistema de información
- **Especificación de requisitos:** es la información detallada del funcionamiento, restricciones del sistema y las necesidades tanto del cliente como del usuario. Esta información involucra los requisitos funcionales y no funcionales del sistema.
- **Requerimientos funcionales:** Los requerimientos funcionales se basan en el funcionamiento que tendrá el sistema, en si describe cuáles será el comportamiento de este ante cualquier tipo de

entrada, es decir son especificaciones de los servicios que el sistema debe otorgar.

- **Requerimientos no funcionales:** Los requerimientos no funcionales establecen especificaciones que llegan a ser las restricciones del comportamiento de la aplicación, por lo general estas especiaciones son otorgados por el sistema.
- **Requerimientos de usuario:** es la información representada en lenguaje natural o diagramas de tal manera que sea entendible para las personas encargadas de manejar el sistema, en ello detallan las funciones y restricciones en las que el sistema operará.

LECTURAS COMPLEMENTARIAS

"Software Engineering: A Practitioner's Approach" de Roger S. Pressman Es un libro que trata todos los aspectos de la ingeniería de software: modelos de ciclo de vida, requisitos, diseño, pruebas, mediciones.

Lo puede encontrar en:

- http://www.librarything.com/work/153862/reviews/3136349

Desarrolle las siguientes preguntas

1. ¿Cuáles pueden ser las consecuencias de una escasa recopilación de requisitos?
2. Establezca una diferencia entre requisitos funcionales y no funcionales
3. Cite 10 ejemplos de requerimientos de usuario
4. ¿Cuál es la consecuencia de no considerar los requisitos funcionales?
5. ¿Por qué son importantes los requisitos no funcionales en el funcionamiento del sistema?
6. ¿En que se basan los requerimientos de dominio?

"Ingeniería y Arquitectura del Software" de Ángel Arias y Alicia Durango es un libro de ayuda para aquellas personas que son ingenieros informáticos.

Lo puede encontrar en:

https://books.google.com.ec/books?id=uADPCgAAQBAJ&dq=libros+para+la+ingenieria+de+software%3B+requerimientos&source=gbs_navlinks_s

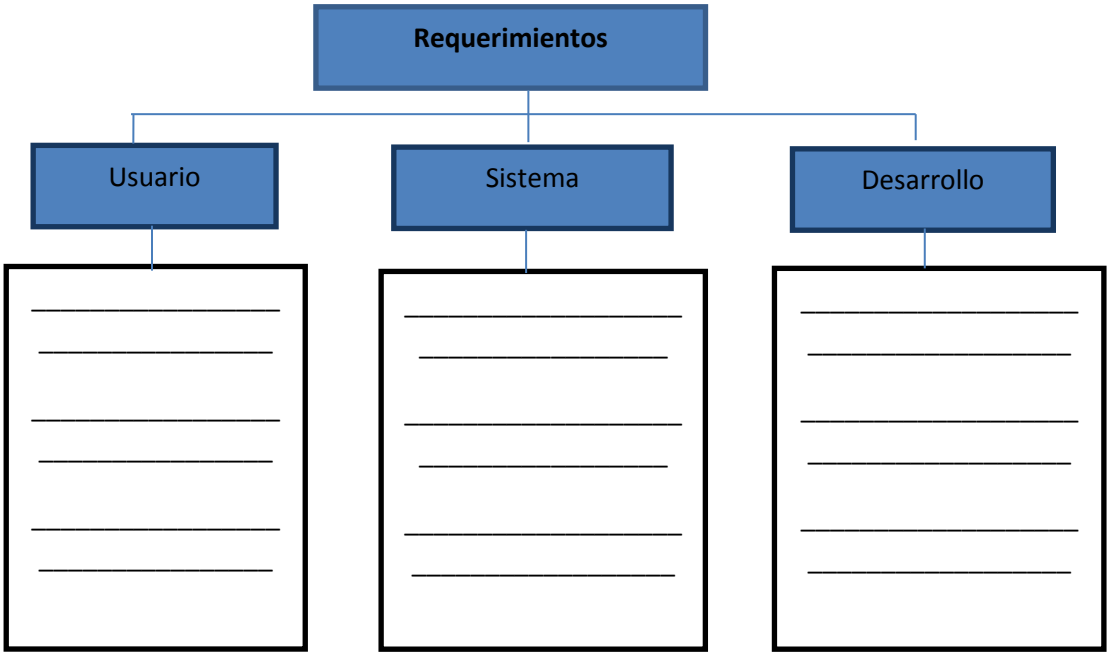
Desarrolle las siguientes preguntas

1. ¿De qué manera ayuda la obtención y análisis de requerimientos?
2. ¿Es importante considerar un documento de requerimientos para la obtención de información? ¿Por qué?
3. ¿Por qué es importante considerar los requerimientos del usuario?
4. ¿Puede ser considerado los requerimientos del sistema como requerimientos funcionales? ¿Por qué?
5. ¿En que se basa el estándar IEEE – 830 – 1998 para el desarrollo de documentos de requerimientos?
6. Si en la estructura establecida en el entandar IEEE para los documentos de requerimientos se omiten ciertos puntos ¿En qué manera afectaría esto en la obtención de los requerimientos?

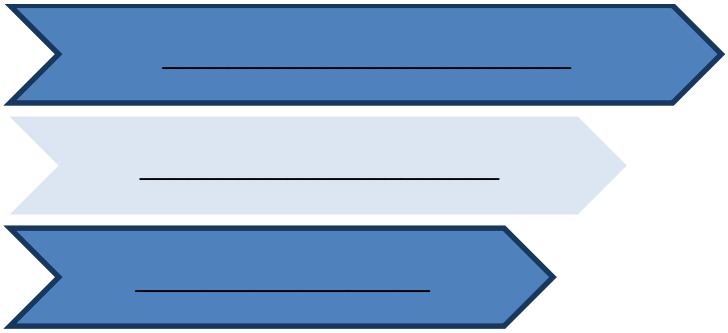
TALLER

TALLER N° 1 – Los requerimientos como parte fundamental de un software.

1. **Realizar un diagrama donde identifique las características de los tres tipos de requerimientos**



2. **¿Cuáles son los tipos de requerimientos no funcionales?**



TALLER N°2 - ERS

1. **Identificar los formatos de especificación de requisitos de software según la norma IEEE 830.**

   _____
   _____
   _____
   _____
   _____
   _____

2. **Describa los conceptos sobre el diseño de contrato utilizados con la norma sobre un software**

   Objeto: _____

   _____

   Modificaciones de requisito: _____

   _____

   Entrega del sistema: _____

   _____

   Penalizaciones: _____

   _____

   Propiedad intelectual: _____

   _____

   Condiciones económicas: _____

   _____

   Garantía: _____

   _____

   Resolución del contrato: _____

_____

General: _____

_____

Sumisión: _____

_____

TALLER N° 3 – Requerimientos

Responda a las siguientes preguntas:

1. **¿Cuál es la importancia de los requerimientos de usuario dentro del desarrollo del proyecto?**

   _____

   _____

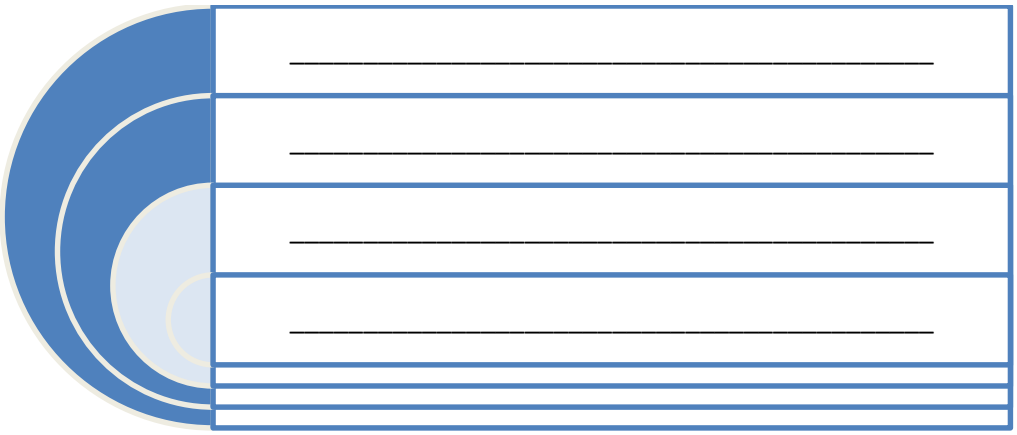
   _____

   _____

2. **Realice un cuadro comparativo entre requerimientos funcionales y no funcionales**

Requerimientos funcionales	Requerimientos no funcionales
• _____	• _____
• _____	• _____
• _____	• _____
• _____	• _____
	• _____

3. **Mencione 4 aspectos de los perfiles de usuario**



EVALUACIÓN

1	**FORMATO**	**SIMPLE**
	Contexto	
	Planteamiento	**Son componentes mínimos en un documento de requisitos de software**
	OPCIÓN A	Alcance del sistema
	OPCIÓN B	Requisitos funcionales
	OPCIÓN C	Historias de usuario
	OPCIÓN D	Requisitos funcionales
	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios
2	**FORMATO**	SIMPLE
	Contexto	
	Planteamiento	**Los requerimientos no funcionales se refieren a:**
	OPCIÓN A	Las declaraciones de los servicios que proveerá el sistema
	OPCIÓN B	Las transformaciones que el sistema realiza sobre las entradas para producir salidas
	OPCIÓN C	La manera en que éste reaccionará a entradas particulares
	OPCIÓN D	Las restricciones de los servicios o funciones otorgados por el sistema
	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios
3	**FORMATO**	SIMPLE
	Contexto	
	Planteamiento	**Quien se encarga de establecer los detalles de los servicios y las restricciones del sistema**
	OPCIÓN A	Requerimientos del sistema

	OPCIÓN B	Requerimientos de usuario
	OPCIÓN C	Requerimientos de dominio
	OPCIÓN D	Requerimientos de desarrollo
	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios
4	**FORMATO**	SIMPLE
	Contexto	
	Planteamiento	**En esta actividad de gestión del proyecto de software se compara el progreso con los costes anuales y los planificados**
	OPCIÓN A	Planificación
	OPCIÓN B	Selección
	OPCIÓN C	Supervisión
	OPCIÓN D	Redacción y presentación de informes
	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios
5	**FORMATO**	SIMPLE
	Contexto	
	Planteamiento	**Son tipos de requerimientos no funcionales, excepto:**
	OPCIÓN A	Fiabilidad
	OPCIÓN B	Seguridad
	OPCIÓN C	Rendimiento
	OPCIÓN D	Usabilidad
	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios
6	**FORMATO**	SIMPLE
	Contexto	
	Planteamiento	**Son características para una buena especificación de requisitos, excepto**
	OPCIÓN A	Eficiente

	OPCIÓN B	Verificable
	OPCIÓN C	Completa
	OPCIÓN D	No ambigua
	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios
	FORMATO	SIMPLE
	Contexto	
	Planteamiento	**¿Cuándo un documento es no ambiguo?**
	OPCIÓN A	No hay diferencia entre ellos
	OPCIÓN B	Todos los requisitos están descritos casi por el mismo término
	OPCIÓN C	si cada requisito descrito tiene una única interpretación
	OPCIÓN D	Si los requisitos tienen similar interpretación
	Respuesta Correcta	
	Nivel	1
7	**Operación Cognitiva**	Aplicación de Conceptos y Principios
	FORMATO	SIMPLE
	Contexto	
	Planteamiento	**¿Un ERS es explorable cuándo?**
	OPCIÓN A	El origen de cada requerimiento es claro tanto hacia atrás como hacia delante
	OPCIÓN B	Tiene en cuenta las necesidades de mantenimiento
	OPCIÓN C	.El personal que no ha intervenido directamente en el desarrollo
	OPCIÓN D	Los requisitos pueden cambiar
	Respuesta Correcta	
	Nivel	1
8	**Operación Cognitiva**	Aplicación de Conceptos y Principios
	FORMATO	SIMPLE
	Contexto	
9	**Planteamient**	**Los perfiles de usuario tienes estos aspectos,**

	o	**excepto:**
	OPCIÓN A	Experiencia en torno a hardware y software
	OPCIÓN B	Experiencia en aplicaciones del mismo
	OPCIÓN C	Experiencia en el trabajo
	OPCIÓN D	Experiencia en el desarrollo de software
	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios
	FORMATO	SIMPLE
	Contexto	
	Planteamiento	**Si existe una definición de respuestas a todas las posibles entradas, tanto válidas como inválidas, en todas las posibles situaciones una ERS es:**
	OPCIÓN A	Consistente
	OPCIÓN B	Completa
	OPCIÓN C	Modificable
	OPCIÓN D	Correcta
	Respuesta Correcta	
	Nivel	1
10	**Operación Cognitiva**	Aplicación de Conceptos y Principios

GLOSARIO

- **Ámbito del Sistema:** "Describirá los beneficios, objetivos y metas que se espera alcanzar con el futuro sistema" [33]
- **Auditoría interna:** "Proceso sistemático, practicado por los auditores de conformidad con normas y procedimientos técnicos establecidos" [34]
- **Estándar:** "Documento establecido por consenso, aprobado por un cuerpo reconocido, y que ofrece reglas, guías o características para que se use repetidamente." [35]

- **Interfaz externa:** "Describirá los requisitos que afecten a la interfaz de usuario, interfaz con otros sistemas (hardware y software) e interfaces de comunicaciones." [33]
- **Requerimiento:** "Especifica qué es lo que el sistema debe hacer (sus funciones) y sus propiedades esenciales y deseables." [37]
- **Requerimientos externos:** "A qué se debe atener el sistema o software con respecto a su entorno: compatibilidad con otros sistemas, adecuación a determinadas leyes, etc." [38]
- **Requerimiento funcional:** "Característica requerida del sistema que expresa una capacidad de acción del mismo – una funcionalidad; generalmente expresada en una declaración en forma verbal." [36]
- **Requisitos futuros: "**Subsección esboza futuras mejoras al sistema, que podrían analizarse e implementarse en un futuro." [33]
- **Requerimiento no funcional: "**Característica requerida del sistema, del proceso de desarrollo, del servicio prestado o de cualquier otro aspecto del desarrollo, que señala una restricción del mismo." [36]
- **Stakeholder:** "Término agrupa a trabajadores, organizaciones sociales, accionistas y proveedores, que se ven afectados por las decisiones de una empresa." [39]

SOLUCIONARIO

1. C) Historias de usuario
2. D) Las restricciones de los servicios o funciones otorgados por el sistema
3. A) Requerimientos del sistema
4. C) Supervisión
5. B) Seguridad
6. A) Eficiente
7. C) Si cada requisito descrito tiene una única interpretación
8. A) El origen de cada requerimiento es claro tanto hacia atrás como hacia delante
9. D) Experiencia en el desarrollo de software
10. B) Completa

4

Paradigma Orientado a Objetos con UML

Competencias

- Selecciona correctamente los componentes de UML.
- Aplica herramientas en el Modelado de Casos de Uso.
- Elabora la matriz de trazabilidad.

Cuando haya leído esta unidad el **resultado de aprendizaje**:

- Aplicar unas herramientas Open Source para modelar el sistema, como StarUML.

Contenidos

4.1 Paradigma Orientados a Objetos y Artefactos UML.
4.2 Diagramas de Caso de Uso.
4.3 Formatos de Diagramas de Casos de Uso.
4.4 Diagramas de Actividad.

Introducción al Capítulo IV

La programación orientada a objetos y la modelación orientada a objetos, hoy en día se han convertido en la mejor técnica para especificar de una forma semiforme y entendible para los paradigmas orientados a objetos.

El lenguaje de modelado unificado mejor conocido como UML (Unified Modeling Language) se ha convertido hoy en día en un lenguaje estandarizado para desarrollar y unificar las diferentes notaciones existentes entre los paradigmas orientados a objetos. El lenguaje UML se ha logrado diseñar para satisfacer todas las necesidades de producción de los paradigmas orientados a objetos desde el punto de vista del usuario y desarrollador indistintamente (hay diagramas desde el punto de vista del usuario y otros del desarrollador), el modelado orientado a objetos tiene la necesidad de comprender, tecnificar y analizar las distintas aplicaciones que contendrán el software a desarrollar y el dominio en el que van a operar.

Dentro de este capítulo se da una breve descripción de lo que es el paradigma orientado a objetos, UML y los diferentes diagramas que pueden intervenir en el UML para el ciclo de desarrollo de vida del software, como se interpreta su diagramación para de ahí pasarlo al diseño y codificación, como llegar a modelar un casos de uso y sus elementos como también una breve descripción de lo que es el diagrama de actividades.

4 Paradigma Orientado a Objetos con UML

Lenguaje de Modelado Unificado o Unified Modeling Language (UML) es un lenguaje estandarizado que le permite al desarrollador construir, modelar, diagramar y documentar todos las partes que contiene un software que este orientado a objetos o POO. Al pasar el tiempo el UML se ha convertido en la herramienta principal de trabajo para el desarrollo de ciclo de vida del software, convirtiéndose en un lenguaje estandarizado. El lenguaje estandarizado UML ha dado fin a las guerras de las metodologías de ciclo de desarrollo de software porque cada metodología puede escoger cualquier diagrama (el que más le favorezca), para el desarrollo de un software; además de esto que ha ayudado a las herramientas CASE a realizar un poco de Ingeniería Inversa cerrando ya los métodos y atributos para el POO.

4.1 Paradigmas Orientados a Objetos y Artefactos UML

Los paradigmas Orientados a Objetos (POO, OOP según sus siglas en español o inglés respectivamente) es la utilización de clases u objetos que ayudan a trazar y relacionar las diversas aplicaciones orientándolas a un paradigma de la programación. El POO se basa en la utilización de varias características como son: abstracción, polimorfismo, acoplamiento, encapsulamiento y el más utilizado que es la herencia entre clases.

Es decir que la POO monopoliza objetos para construir un sistema o software utilizando estilos de programación.

4.1.1 Paradigma Orientado a Objetos

El Paradigma Orientado a Objetos (POO) hoy en día se ha convertido en la mejor forma para crear un software, es decir se ha creado como una filosofía para el desarrollo e implementación de un software porque le permite al desarrollador crear dispositivos, funcionales agrandables y genéricas, de modo que el cliente o usuario final pueda utilizar el software de la manera que más le convenga.

El paradigma orientado a objetos se centra en la utilización principal de sus tres características principales, las cuales son: La abstracción de datos, la identidad de objetos y la más utilizable que es la herencia; aunque cabe resaltar que existen otras características de la POO las cuales son:

- **Estructura modular basada en objetos:** Todos los datos del sistema para el desarrollador del software deben ser modularizada de acuerdo a la estructura que necesite el software.
- **Abstracción de datos:** Es una característica principal de la POO que se ha establecido como una técnica que represente por completo toda la estructura de los datos de un software, minimizando la complejidad del software para el usuario final, es decir minimizar la complejidad del sistema para el usuario.
- **Gestión automática de memoria:** Esta característica establece la autonomía para el software, es decir que el software sea capaz de desasignar objetos a su voluntad mientras estos no sean requeridos por el usuario sin intervención del equipo de desarrollo.
- **Clase:** Plantilla para la creación de objetos, contiene métodos y atributos de cada uno de ellos.
- **Herencia:** Una subclase contiene todos los atributos y métodos de una superclase.

- **Polimorfismo y enlace dinámico:** Un método puede comportarse de diferentes maneras y en diferentes clases.
- **Herencia múltiple:** Una clase puede heredar atributos y métodos de diferentes superclase.

La POO alienta a los desarrolladores a crear sistemas o software más estructurados en vez de pensar en sus procedimientos, el fin del paradigma orientado a objetos es que un objeto conjuntamente con sus atributos y métodos puedan emitir mensajes al software para que estos desarrollen una acción específica que solucionen un inconveniente.

Características de la P.O.O.

- **Abstracción:** Es donde cada objeto dentro del software se lo utiliza como un modelo abstracto para que realice un trabajo determinado, es decir que el objeto cambia el estado con otros objetos sin dejar ver como este se implementa.
- **Encapsulamiento:** El encapsulamiento dentro de la POO es que un objeto se aísle del exterior y logre exponer una interfaz a otros objetos que específica cómo pueden interactuar con los elementos de la clase. Además que el encapsulamiento protege cada propiedad de un objeto a ser manipulado o cambiado por un agente externo, es decir que no cambien las propiedades del objeto encapsulado.
- **Polimorfismo:** El polimorfismo es el comportamiento distinto que pueden tener los métodos dentro de una clase ya que se puede establecer el mismo nombre del método pero tiene la opción de poseer un comportamiento diferente. Dentro del polimorfismo existe también el bien llamado "tiempo de ejecución" que es la línea de vida de un objeto en sus diferentes comportamientos.
- **Herencia:** La herencia es nada menos que una clase hija hereda sus atributos y métodos de su clase padre; es decir que cada clase que compone un sistema o software puede o está relacionada entre sí. Para la utilización de la herencia dentro de un software se necesita que se cuente con el encapsulamiento y polimorfismo para que un objeto logre ser definido y creado. En la actualidad existen también herencia múltiple que es soportado solo por unos tipos de lenguajes de programación que es donde un objeto

hereda sus métodos y atributos de diferentes clases; pero esto no es soportado por uno de los lenguajes más conocido como java.

Para poder aplicar los paradigmas orientados a objetos, es necesario que el desarrollador conozca y defina los elementos principales; los cuales son:

- **Objeto:** es la unió de atributos y métodos que pertenecen a un objeto del mundo real, un objeto define el comportamiento y propiedades de una clase.
- **Clase:** Plantilla para la creación de objetos, contiene métodos y propiedades.
- **Método:** Los métodos dentro de la POO son simplemente el comportamiento de un objeto donde por publicación deben devolver un "mensaje"; es decir un método es lo que puede hacer un objeto.
- **Evento:** Es el mínimo acontecimiento dentro de un programa; ya sea una interacción del software con el usuario final o simplemente un recado enviado por un objeto hacia otro objeto.
- **Mensaje:** El mensaje dentro de la POO es la comunicación que se crea hacia un objeto dentro de una clase; es decir es la orden de ejecución de un método.
- **Atributo:** Son las propiedades de los objetos donde los valores de un objeto pueda ser alterado solo por la ejecución de algún comportamiento del objeto.
- **Estado interno de un objeto**: es una propiedad invisible de ellos; que para poder ser autorizada o cambiada debe de haber por medio un método dentro del objeto que establezca el permiso de cambio.

4.1.2 Herramientas CASE

"Las herramientas CASE ponen a su alcance un medio para comunicar aspectos del sistema durante su conceptualización. A través de apoyo automatizado que incluye salida en pantalla, los clientes pueden apreciar de inmediato cómo están representados los flujos de datos y otros conceptos del sistema, y pueden solicitar correcciones o cambios que hubieran tomado demasiado tiempo con herramientas anteriores". [40]

"Las herramientas CASE se pueden utilizar para generar un programa esqueleto a partir de un diseño. Esto incluye código para definir e implementar las interfaces, y en muchos casos el

desarrollador solo necesita agregar detalles del funcionamiento de cada componente del programa". [41]

Una herramienta CASE es una herramienta que ayuda a modelar Lenguaje Estandarizado UML para solucionar problemas de la Ingeniería de software; una herramienta CASE puede incluir una diversidad de herramientas informáticas que ayudan a maximizar la productividad en el ciclo de desarrollo de un software, reduciendo costo y tiempo; Las herramientas CASE además de esto pueden generar código, a partir del trazo de un diagrama, en un lenguaje de programación previamente establecido.

4.1.3 ¿Qué es UML?

Unified Modeling Language (UML), que significa lenguaje de modelado unificado, hoy en día es un lenguaje estandarizado utilizado en casi todas las metodologías de desarrollo del ciclo de vida de un software. Es una notación que utiliza diagramas, cuadros, entre otras cosas para poder construir, visualizar y documentar todo lo necesario (requisitos del sistema) para el desarrollo de un software, el lenguaje UML es un grupo de aplicaciones para la fabricación de sistemas comerciales y no comerciales.

"Por diseño, UML no es una metodología. Aunque los iniciadores y autores de UML reconocen la importancia de las metodologías, que consideran el marco específico y las condiciones de un dominio de aplicación, el ambiente organizacional, y muchas otras cosas. UML se puede utilizar dentro de las diversas metodologías y puede constituir la base de diversos enfoques, ya que proporciona un conjunto definido de constructores de modelado, junto con la notación uniforme y semántica" [42].

Para poder entender lo que es UML primero debemos referirnos que UML es un lenguaje, es decir que contiene una sintaxis y sobretodo una semántica. Y para poder entender cómo se debe modelar UML es necesario conocer las reglas de oro, los elementos que influyen en él y su significado; pero ante todo esto también se debe conocer cómo se debe ordenar UML y en que metodología se encuentra presente una parte de esta.

UML hoy en día no solo llega a representar la parte conceptual de un sistema si no que ayuda a definir el contexto de cómo tratar al sistema para su construcción

UML sirve generalmente para:

- Desarrollo de software comercial y no comercial.
- Desarrollo de software de proyectos.

- Ayuda a capturar los requerimientos para la creación de un sistema.
- Y sobre todo para la documentación ya sea de un sistema nuevo o uno existente.

4.1.4 Repertorio de artefactos de modelado UML 2.0

UML 2.0 trata de manejar cada artificio es decir las entidades físicas del sistema y desplegarlas a los diferentes nodos, que son los dispositivos y ambientes de ejecución.
UML 2.0 permite elaborar diferentes tipos de diagramas, añadiendo a este conjunto los diagramas de estructura compleja, de interacción y de tiempo.

4.2 Diagramas de Casos de Uso

En UML los diagramas de Caso de Uso significan el comportamiento que debe seguir el software o el sistema en general, este es especificado desde el punto de vista de los usuarios; Los diagramas de caso de uso se definen como una notación gráfica.
Los diagramas de caso de uso desafían la tarea de negocio. Un diagrama de caso de uso describe cada especificación completa y suplementaria para encarcelar los requerimientos del sistema, los diagramas de caso de uso nacen de los requisitos que debe tener el sistema.
En esta parte del capítulo se da una breve descripción de qué son los diagramas de caso de uso, cuál es su importancia, sus características y sus partes.

4.2.1 Casos de Uso – Importancia

La importancia del caso de uso es que un caso de uso redacta todas las funciones básicas o compuestas del sistema desde el punto de vista de los usuarios internos y externos de manera que estos usuarios puedan entenderlos, los casos de uso describen el comportamiento de los usuarios por lo tanto es de vital importancia que al momento de crearlos el usuario final debe revisar y aceptar el caso de uso.
"Un caso de uso representa un objetivo sencillo de un sistema y describe una secuencia de actividades y de interacciones con el usuario para alcanzar el objetivo. Los casos de uso proporcionan una sólida base para el desarrollo de manuales y sistemas de ayuda para los usuarios, así como para la creación de documentación sobre el desarrollo del sistema [43]
Para lograr crear diagramas de casos de uso, se debe comprender el sistema a partir de las situaciones y servicios que debe ofrecer

el sistema de trabajo en su entorno; los casos de uso son independientes de los aparatos que se introduzcan dentro del sistema para que este funcione.

4.2.2 Elementos Básicos: Actor, Caso de Uso

Los elementos principales que se deben entender para poder crear un diagrama de caso de uso son:

Actor: los actores son las personas o máquinas encargadas de interactuar con el sistema ya sean internas o externas dentro de la organización; un actor también puede ser la unión de personas informes al sistema o simplemente máquinas que vayan a interactuar con el software. Por ejemplo dentro de un sistema de inventario y facturación se van a encontrar con personas como son el bodego o vendedor, estas personas nos conlleva a que son los actores internos del sistema; pero los actores externos del sistema son los clientes y proveedores.

Para poder elaborar correctamente un diagrama de caso de uso es importante aclarar la diferencia que existe entre un actor a un usuario; un actor corresponde a un rol que va a tener una persona o maquina dentro del software mientras que el usuario es la persona que, cuando usa el sistema, asume un rol. Para poder entender esto se toma el ejemplo de que dentro de un sistema operativo cualquiera, donde un usuario puede crear diferentes perfiles; es decir que un usuario puede acceder al sistema en diferentes perfiles del sistema operativo, donde nos lleva que los perfiles son equivalentes a las funciones del actor que pueden hacer funciones distintas.

Caso de Uso: Un caso de uso es la descripción de las actividades que se deben realizar en un software; un caso de uso siempre es inicializado por un actor, el caso de uso siempre describe un servicio que debe dar el software a crear.

Los casos de uso se representan por medio de una elipse y dentro de la elipse se encuentra el nombre del caso de uso que debe estar denotado por un verbo en gerundio como “Crear”, “Modificar”.

La importancia de los casos de uso es que siempre se expresan desde el punto de vista del actor y no deben de estar denotados desde el punto de vista del sistema.

“Las características de los casos de uso Según Ceria son:

1. Están expresados desde el punto de vista del actor.
2. Se documentan con texto informal.

3. Describen tanto lo que hace el actor como lo que hace el sistema cuando interactúa con él, aunque el énfasis está puesto en la interacción.
4. Son iniciados por un único actor.
5. Están acotados al uso de una determinada funcionalidad –claramente diferenciada– del sistema". [44]

4.2.3 Descripción de un caso de Uso – Notación Gráfica

Para poder describir un caso de uso el desarrollador debe entender que estos se detallan o describen en un texto informal y por lo general se numera los pasos necesarios para que el sistema interactúe con el actor. A continuación se muestra como se debe describir un diagrama de caso de Uso

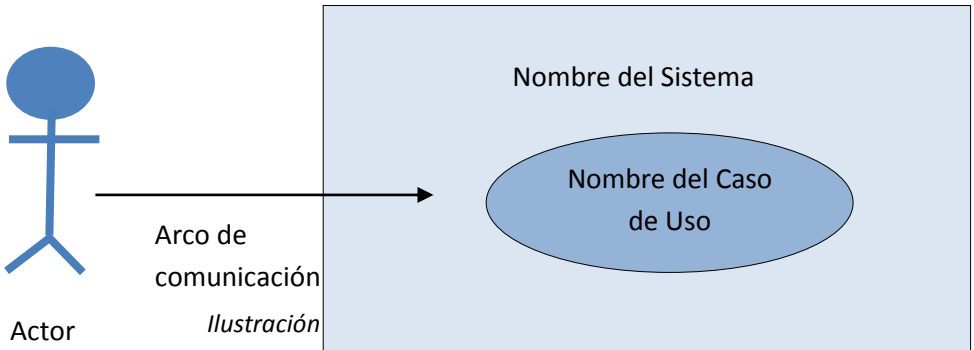
Caso de Uso: Nombre del Caso de Uso
Actor: Las personas o maquinas que van a interactuar con el sistema
1. descripción del servicio que debe dar el sistema documentado de forma informal
2. descripción del servicio que debe dar el sistema documentado de forma informal

Tabla 5 Descripción Simple de Caso de Uso

Caso de Uso: Ingreso Pedido
Actor: Empleado de Ventas
1. EL cliente se comunica con la oficina de ventas, e informa su número de cliente
2. el oficial de ventas ingresa el número de cliente en el sistema
3. EL sistema obtiene la información básica sobre el cliente
4. el cliente informa del producto que quiere comprar, indicando la cantidad
5. El sistema Obtiene la información sobre el producto solicitado y confirma su disponibilidad
6. Se repite el paso 4 hasta que el cliente no informa más productos
7…

Tabla 6 Descripción Simple de Caso de Uso "Ingresar Pedido"

Una vez terminado la descripción del caso de uso se procede a graficarlo:
Un caso de uso corresponde a un grafo con dos tipos de nodos que son el actor y el caso de uso pero en medio de ellos hay una línea de comunicación (flecha) que es el flujo de comunicación.
"Un Actor representa cualquier elemento que intercambia información con el sistema, por lo que está fuera de él. En cambio un Caso de Uso es una secuencia de intercambios en un dialogo con el sistema que se encuentran relacionadas por su comportamiento" [45].



Ilustración

Los casos de uso siempre se deben representar utilizando una elipse u ovalo y en el centro el nombre del caso de uso, el caso de uso representa el requisito del sistema que se debe implementar para solucionar un problema. Un caso de uso representa un trabajo completo que deben de desarrollar los actores del sistema.

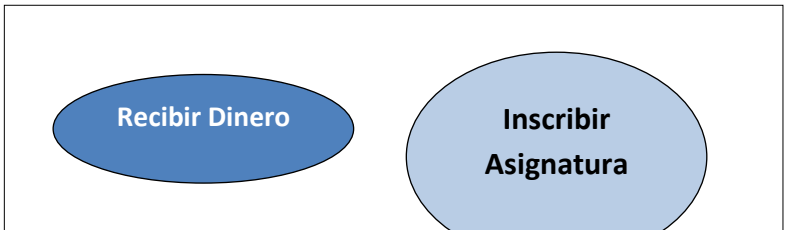


Ilustración 7 Ejemplo Caso de Uso

Para pc ɔ de uso generalmente es representado por el dibujo simple de una persona, donde debajo de ella va la función o el nombre del actor

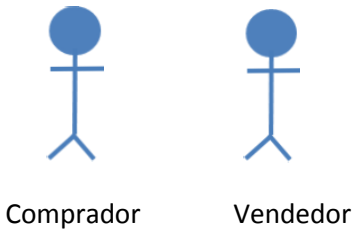
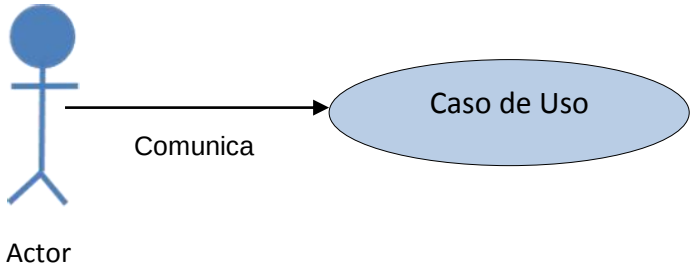


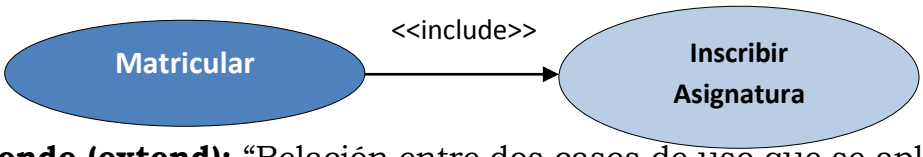
Ilustración 8 Ejemplo Actor

La notación de la comunicación entre el actor y el caso de uso es conocida como Relaciones de un Diagrama de Caso de Uso, las relaciones de un caso de uso se dividen en:

Comunica (comunicantes): esta es la relación más común y sol representa que un actor está relacionado directamente con un caso de uso determinado.



Use (incluye): Esta comunicación existe solo entre dos casos de uso y rep *Ilustración 9 Relación Comunicaste* dentro de otro caso de uso; es decir que para que se cumpla un caso de uso debe darse obligatorio el caso de uso que se está señalando.



Extiende (extend): "Relación entre dos casos de uso que se aplica cuand *Ilustración 10 Relación Include* or ejemplo, podría tenerse un caso de uso que extienda la forma de pago a un cajero, la que podría ser al contado o con cheque" [40].

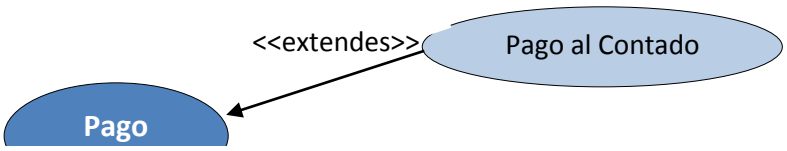


Ilustración 11 Relación Extend

4.2.4 Caso de Uso y Esc os Organizacion

Los escenarios organizac la unión y secuencia de actividades entre el actor y los casos de uso (acciones del sistema) que ayudan a describir cómo interactúan entre ambos; los escenarios clásicos que encontramos son:

- **Escenario básico:** El escenario básico significa que todo se encuentra en perfecto estado
- **Escenario alternativo.-** Significa que se encontraron funciones excepcionales o errores que se deben corregir,

comúnmente se los llama fallos del sistema y las excepciones fallos de validación

4.3 Formatos de Diagramas de Casos de Uso

Los diagramas de caso de uso nos muestran una forma simple de cómo debe interactuar el sistema con los diferentes actores, para poder detallar más el caso de uso se necesita de un formato que muestre las especificaciones de lo que hace el caso de uso.

Los formatos de caso de uso son los resúmenes de las reacciones que se dan entre los diferentes caso de uso, nos muestra el orden de ejecución y el objetivo de este. Este documento o formato del diagrama de caso de uso debe estar vinculado al caso de uso a mostrar.

El formato de los diagramas de caso de uso es la caja negra que suministra el caso de uso, esta caja negra debe de mostrar cómo trabaja paso a paso el caso de uso; debe mostrar el flujo normal y el flujo excepcional del sistema.

"Las interacciones se expresan en diagramas de secuencia, colaboración y actividad, así, hay un enlace entre las vistas funcional y dinámica del sistema y las clases en la implementación de los casos de uso se modelan y se describen en diagramas de clases y de estados" [46].

4.3.1 Diagramas de Caso de Uso - Elementos

Actor: "Un actor es un elemento externo que interacciona con el sistema de información, los actores son los encargados de iniciar los casos de uso que representan las actividades que el sistema de información debe realizar" [47].

Un actor es representado por una persona o un objeto externo (fuera del sistema), que interactúa con el sistema (normalmente iniciándolo) dentro de un caso de uso.

Un actor no solo representa a una persona u objeto un actor debe representar obligadamente un rol dentro del sistema, por ejemplo, un persona representa a un vendedor dentro de una empresa de ventas de productos de primera necesidad, donde esta persona realiza los pedidos de los clientes, este vendedor representa el rol de equipo de ventas.

Una definición clara de un actor es que es la representación de un rol con respecto a un sistema. El rol del actor es la labor que debe realizar dentro del software.

Dentro de los diagramas de caso de uso existen 4 diferentes tipos de actores los cuales son:

- **Actores primarios de negocio:** Son las persona que de alguna forma obtienen algún beneficio de la función del sistema
- **Actores primarios del sistema:** Las personas u objetos que se relacionan directamente con el sistema
- **Actores de servicio externos:** son los actores externos que necesitan responder la petición del caso de uso, por ejemplo un cliente
- **Actores de recepción externo:** este último actor tiene la caracterización de no ser un actor primario pero sin embargo recibe algún beneficio del sistema. Por ejemplo el empleado que puede tener un actor de recepción externo que es el tiempo.

Casos de Uso: "Describen funciones básicas o simples desde la perspectiva del usuario externo y de manera que él lo pueda comprender" [47]

Relaciones

Asociación: "Es el tipo de relación más básica que indica la invocación desde un actor o caso de uso a otra operación (caso de uso). Dicha relación se denota con una flecha simple" [43].

Las asociaciones es la relación más conocida y simple del caso de uso y representa la unión entre el actor y el caso de uso; la asociación es representada por una línea solida apuntando con una saeta al caso de uso.

Dependencia o Instanciación: la dependencia o instanciación es la necesidad con la que cuenta un caso de uso hacia otro; es decir que un caso de uso no se puede cumplir sin que el otro caso de uso ya se haya cumplido.

Generalización: la generalización es la relación más utilizada dentro de los diagramas de caso de uso porque cumple con una doble función dependiendo de su necesidad, estas pueden ser de uso (<<uses>>) que es recomendable utilizarla cuando se tiene un conjunto de características que son similares en más de un caso de uso y no se desea mantener copiada la descripción de la característica. Y la otra es la de Herencia conocido como extends esta relación de generalización se la utiliza cuando un caso de uso tiene las mismas características al otro.

4.3.2 Formato de Alto Nivel

El formato de alto nivel es una descripción rápida de todos los procesos que realiza el diagrama de caso de uso; el formato de alto nivel está dado por el siguiente esquema:

Casos de Uso: Nombre del Caso de Uso **Actores:** Nombre de los actores **Tipo:** 1. Primario, Secundario y opcional (para discutirlo). O 2. Esencial o real (para discutirlo). **Descripción:** Narración de lo que hacen los actores. Se sugiere el siguiente esquema: Este caso de uso inicia cuando <actor> <Inicia un EVENTO>.

Tabla 7 Formato de Alto Nivel

4.3.3 Formato Expandido

El formato Expandido en cambio muestra a detalle cada paso del diagrama de caso de uso, este formato se lo utiliza para obtener una mejor comprensión de los procesos y requisitos del sistema.
El formato expandido cuenta con:

- **Parte Superior:**
  - **Caso de uso:** Nombre del Caso de Uso
  - **Actores:** Lista de actores, indicando quién inicial el caso de uso.
  - **Propósito:** Intención del caso de uso
  - **Vista General:** La misma descripción del caso de uso de alto nivel o algún resumen similar.
  - **Tipo:**
    1. Primario, secundario u opcional (para discutir).
    2. Esencial o real (para discutir).
  - **Referencia cruzada:** Funciones del sistema y casos de uso relacionados.
- **Sección Media o corazón**
  Curso típico de eventos en términos de cada una de las acciones del actor y la correspondiente respuesta del sistema
- **Sección Final**
  Curso de eventos alternativos (describe alternativas importantes o excepciones que pueden presentarse respecto al curso típico).

Por ejemplo:

Caso de uso: Registrar vehículo

Actores: Usuario

Propósito: Registrar un vehículo en el sistema.

Tipo: Primario

Descripción: El usuario registra un vehículo a una empresa en el sistema rellenando los datos del vehículo. El sistema comprueba si el vehículo existe. Si no existe registra el vehículo en el sistema.

Curso normal de los eventos:

1. El usuario introduce los datos del vehículo que quiere registrar.

2. El sistema muestra una lista de empresas registradas en el sistema

3. El usuario selecciona una empresa.

4. El sistema busca la empresa seleccionada y muestra los datos de la empresa.

5. El sistema busca en los vehículos registrados si el vehículo ya está registrado en el sistema.

6. El sistema registra el vehículo en la empresa y muestra un mensaje satisfactorio al usuario.

Cursos alternos:

Línea 6: El sistema no registra el vehículo y muestra un mensaje informando al usuario que el vehículo ya está registrado.

Tabla 8 Formato Expandido

4.3.4 Matriz de Trazabilidad

"La Matriz de Trazabilidad designa la capacidad de rastrear la estructura de relaciones entre los requisitos, a modo de comprender los impactos que el cambio en un requerimiento específico pueda provocar en el desarrollo de un sistema este tipo de matriz cruza los requisitos entre si y verifica que interfieren entre sí" [48]

La matriz de trazabilidad contiene información entre los requerimientos del sistema para poder encontrar así su dependencia y los componentes del diseño.

La Matriz de Trazabilidad se puede trazar entre:

- Los requisitos del sistema
- Requisitos con caso de uso

4.4 Diagramas de Actividad

Los diagramas de actividades muestran una línea de actividades ordenadas, donde una actividad es la ejecución entre los objetos en un momento dado, este diagrama debe mostrar el proceso de software o el proceso de negocio como un flujo de trabajo mediante acciones de procesos. Los procesos de negocio pueden interpretar:

- Un proceso de negocio o un flujo de trabajo entre los usuarios y el sistema
- Un protocolo de software, esto es la secuencia de pasos entre las actividades permitida.
- Los procesos muestran el algoritmo del sistema a crear

Desde el punto de vista general el diagrama de actividades enseña el flujo de actividades entre las diferentes clases con la finalidad de dar mejor el control del sistema, en un diagrama de actividades deben mostrar las acciones y actividades de distintas clases.

4.4.1 Software de modelado

El software de modelado es una técnica que permite establecer la complejidad de un determinado sistema, es de gran ayuda para los desarrolladores ya que éstos deben tener nociones de cómo tratar al sistema para su construcción. Además se puede utilizar modelos de abstracción que se necesitan para comunicarse con el cliente.

Para lograr utilizar un sistema de software de modelado es de vital importancia utilizar herramientas de modelado y de ingeniería de software automatizada que pueden ayudar a verificar la corrección del modelo.

4.4.2 Diagramas de Actividad

"Los diagramas de actividades se pueden considerar una variedad tanto del diagrama de estados como los diagramas de interacción, ya que sirve para describir los estados de una actividad, que es un conjunto de acciones en secuencia y/o concurrentes en el cual intervienen clasificados" [49]

Los elementos de los diagramas de actividades son:

Estados de actividad y estados de acción: Se representan por medio de un rectángulo pero contiene las puntas

redondeadas, donde en su interior viene representado por la actividad que se va a realizar.

Flujo de objetos: Trata de que un objeto puede cambiar su estado en una acción y a la vez puede ser utilizado para otras actividades.

Flujos simultáneos: Representa cuando dos acciones se ejecutan simultáneamente en una actividad, también es conocido como flujo en paralelo

Flujos de datos: Solo describe una acción hacia otra acción.

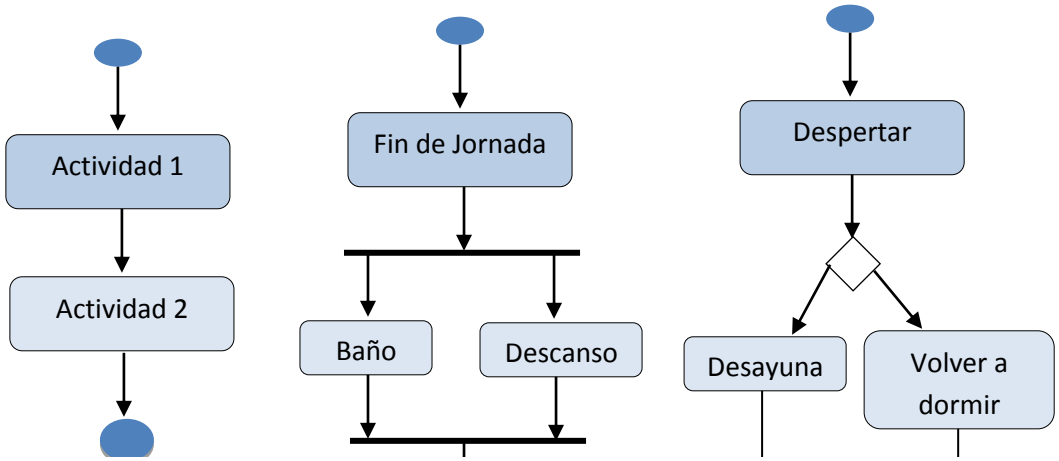


Ilustración 12 Estados de actividad y estados de acción

CONCEPTOS IMPORTANTES

- **Abstracción:** Es donde cada objeto dentro del software se lo utiliza como un modelo abstracto para que realice un trabajo determinado.
- **Actor:** Los actores son las personas o maquinas encargadas de interactuar con el sistema ya sean internas o externas dentro de la organización.
- **Caso de uso:** Un caso de uso es la descripción de las actividades que se deben realizar en un software; un caso de uso siempre es inicializado por un actor, el caso de uso siempre describe un servicio que debe dar el software a crear.
- **Diagramas de actividades** Se pueden considerar una variedad tanto del diagrama de estados como los diagramas de interacción, ya que sirve para describir los estados de una actividad, que es un conjunto de acciones en secuencia y/o concurrentes en el cual intervienen clasificados.
- **Encapsulamiento:** Dentro de la POO es que un objeto se aislé del exterior y logre exponer una interfaz a otros objetos que específica cómo pueden interactuar con los objetos de la clase.

- **Herencia:** Es nada menos que una clase hija hereda sus atributos y métodos de su clase padre
- **Paradigmas Orientados a Objetos:** (POO, OOP según sus siglas en inglés) es la utilización de clases u objetos que ayudan a trazar, relacionar las diversas aplicaciones orientándolas a un paradigma de la programación.
- **Polimorfismo:** Es el comportamiento distinto que pueden tener los objetos dentro de una clase
- **UML 2.0:** Trata de manejar cada artificio es decir las entidades físicas del sistema y desplegarlas a los diferentes nodos, los nodos son los dispositivos y ambientes de ejecución.
- **Unified Modeling Language:** Conocido más como UML es un lenguaje estandarizado que le permite al desarrollador construir, modelar, diagramar y documentar todos las partes que contiene un software que este orientado a objetos o POO.

LECTURAS COMPLEMETARIAS

Del articulo académico: "**Generación del diagrama de casos de uso a partir del lenguaje natural o controlado: una revisión crítica**" de ZAPATA CARLOS; TAMAYO PAULA, disponible en:

- http://www.redalyc.org/articulo.oa?id=49611945021

Desarrolle las siguientes preguntas:

1. ¿Cómo se define Unified Modeling Language UML?
2. ¿Cuáles son las principales ventajas de utilizar los casos de uso según Firesmith?
3. ¿Cuáles son los problemas que se deben resolver para planificar un correcto Diagrama de caso de uso?
4. ¿Cuál es la problemática de la elaboración automática o semiautomática del diagrama de casos de uso?
5. ¿Cuáles son los procesos para la elaboración de un diagrama de caso de uso según García et al?
6. ¿Cuál es la estructura para construir un grafo BRCG desde el código fuente, especifique sus pasos?

Del articulo académico: "**Especificación de Requisitos según el estándar de IEEE 830**" de Gonzalo Méndez, disponible en:
https://www.fdi.ucm.es/profesor/gmendez/docs/is0809/ieee830.pdf

Desarrolle las siguientes preguntas:

1. ¿Se debe seguir estrictamente el formato IEEE 830 o puede ser sujeto a cambios?

2. ¿Se debería especificar un producto totalmente independiente en el documento ERS?
3. ¿Qué se debe colocar en las Funciones del Producto?
4. ¿En qué parte se describen los requisitos que afectan a la interfaz de usuario, interfaz con otros sistemas e interfaz de comunicación?
5. ¿EL documento ERS debería ser legible solo para desarrolladores de software o para personas de muy distintas formaciones e intereses? Justifique su respuesta

TALLER

Taller N° 1 – UML

1. ¿Cuál es la utilidad del diagrama de casos de uso?

_____
_____
_____

2. ¿Cuáles son las relaciones que se pueden presentar entre los casos de uso?

_____
_____
_____
_____
_____

3. Indique como se representa gráficamente las siguientes partes del caso de uso

Actor	
Caso de Uso	
Sistema	

4. Describa el principal caso de uso que genera la máquina de gaseosas.

5. Elabore un listado de las interacciones que se presentan entre sistema y usuario en los siguientes casos: Retiro de dinero por cajero automático

Taller N° 2 – Normas IEEE 830

Conteste las siguientes preguntas acerca de la norma IEEE 830-1998 para informes de ERS (Especificación de Requisitos de Software, o SRS por sus siglas en inglés de Software Requirements Specification).

1. ¿La norma es obligatoria en su estructura o contenido o ambas? Explique

_____
_____
_____
_____
_____

2. ¿Esta norma solo se aplica a sistemas diseñados para Programación Estructurada o para Programación con Orientación a Objetos? Justifique

_____
_____
_____
_____
_____

3. ¿Qué estructura usaría para presentar las funciones del sistema?

_____
_____
_____
_____
_____

4. ¿Cuáles son las características ideales de un requisito (requerimiento) de software?

_____
_____
_____
_____
_____
_____
_____
_____

5. ¿Qué se especifica en la subsección de Requerimientos Lógicos de la Base de Datos?

_____
_____

Taller Nº 3 UML- Normas IEEE - 8.30

1. En la sección de atributos del sistema se detallan ciertos atributos de calidad (requisitos - no funcionales): Fiabilidad, Disponibilidad, Seguridad, Mantenibilidad y Portabilidad. Defina cada uno.

Fiabilidad:

_____

_____

_____

_____

Disponibilidad

_____

_____

_____

_____

Seguridad

_____

_____

_____

_____

_____

Mantenibilidad

_____

_____

_____

_____

Portabilidad

_____

_____

_____

_____

_____

2. ¿Qué define el Alcance (Ámbito o en inglés Scope) del Sistema?

_____

_____

_____

_____

3. Mencione según su criterio 2 posibles problemas del Overscoping en proyectos de software.

_____

_____

_____

_____

4. ¿Qué significado tiene para usted los diagramas de actividades?

_____

_____

_____

_____

5. Bajo su criterio elabore un caso de uso sobre críticos de comida

EVALUACIÓN

	FORMATO	**SIMPLE**
	Contexto	
	Planteamiento	**Que significa UML**
	OPCIÓN A	Unified Modeling Language
	OPCIÓN B	Language Unified Modeling
	OPCIÓN C	Unified Standar Language
	OPCIÓN D	Unified Standar Modeling
	Respuesta Correcta	
	Nivel	1
1	**Operación Cognitiva**	Aplicación de Conceptos y Principios
	FORMATO	SIMPLE
	Contexto	
	Planteamiento	**Que significa POO**
	OPCIÓN A	Paradigma Orientado a Eventos
	OPCIÓN B	Paradigma Orientado a Objetos
	OPCIÓN C	Paradigma Orientado a Acciones
	OPCIÓN D	Paradigma Orientado a Casos de Uso
	Respuesta Correcta	
	Nivel	1
2	**Operación Cognitiva**	Aplicación de Conceptos y Principios
	FORMATO	SIMPLE
	Contexto	
	Planteamiento	**Cuáles son los 4 tipos de requisitos**
	OPCIÓN A	Requisitos de usuarios, sistemas, funcionales y de apariencia
	OPCIÓN B	Requisitos funcionales, no funcionales de usuario y de capacidad
3	**OPCIÓN C**	Requisitos de casos de uso, expandidos, simples y

		funcionales
	OPCIÓN D	Requisitos de usuarios, sistemas, funcionales y no funcionales
	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios
4	**FORMATO**	SIMPLE
	Contexto	
	Planteamiento	**Que muestra el documento ERS**
	OPCIÓN A	Muestra todos los casos de uso
	OPCIÓN B	Muestra los formatos expandidos de los casos de uso
	OPCIÓN C	Muestra las especificación de requisitos de software
	OPCIÓN D	Muestra los formatos simples de los casos de uso
	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios
5	**FORMATO**	SIMPLE
	Contexto	
	Planteamiento	**Entre que se puede trazar la matriz de trazabilidad**
	OPCIÓN A	Entre los requisitos y los requisitos con los casos de uso
	OPCIÓN B	Entre los requisitos y los diagramas de actividades
	OPCIÓN C	Entre los casos de uso y los diagramas de actividades
	OPCIÓN D	Solo entre los requisitos
	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios
6	**FORMATO**	SIMPLE
	Contexto	

	Planteamiento	**Por cual estándar está definido el documento ERS**
	OPCIÓN A	EL documento ERS están definidas por l estándar IEEE 930-1998
	OPCIÓN B	EL documento ERS están definidas por l estándar IEEE 830-1999
	OPCIÓN C	EL documento ERS están definidas por l estándar IEEE 830-1998
	OPCIÓN D	EL documento ERS están definidas por l estándar IEEE 830-1998
	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios
7	**FORMATO**	SIMPLE
	Contexto	
	Planteamiento	**Cuál de las siguientes características pertenece a un caso de uso real.**
	OPCIÓN A	Historias de caso de utilización de un sistema
	OPCIÓN B	Incluye varios paso o transacciones
	OPCIÓN C	Secuencia de interacciones entre un sistema y alguien o algo que usa alguno de sus servicios
	OPCIÓN D	Se documenta con texto informal
	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios
8	**FORMATO**	SIMPLE
	Contexto	
	Planteamiento	**En un sistema de ventas "Supervisor de ventas" representa a:**
	OPCIÓN A	Identificar actores
	OPCIÓN B	Identificar metas
	OPCIÓN C	Identificar los casos de uso a partir de las metas
	OPCIÓN D	Identificar un caso de uso
	Respuesta Correcta	
	Nivel	1

	Operación Cognitiva	Aplicación de Conceptos y Principios
9	**FORMATO**	SIMPLE
	Contexto	
	Planteamiento	**En caso de uso expandido : Llenar actas con notas de alumnos : La descripción :"El profesor no tiene actas en blanco, presenta un mensaje de error y regresa al menú principal "pertenece a :**
	OPCIÓN A	Curso normal de eventos
	OPCIÓN B	Curso alterno de eventos
	OPCIÓN C	Parte Intermedia del Caso de Uso expandido
	OPCIÓN D	Parte Superior del Caso de Uso expandido
	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios
10	**FORMATO**	SIMPLE
	Contexto	
	Planteamiento	**Cuáles son los elementos de un caso de uso**
	OPCIÓN A	Actor, Caso de Uso, acciones
	OPCIÓN B	Actor, eventos, relaciones
	OPCIÓN C	Actor, Casos de Uso y relaciones
	OPCIÓN D	Actor, Dependencias y Casos de uso
	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios

GLOSARIO

- **Actor:** Elemento externo que interacciona con el sistema de información, los actores son los encargados de iniciar los casos de uso que representan las actividades que el sistema de información debe realizar

- **Caso de uso:** Descripción de las actividades que se deben realizar en un software; un caso de uso siempre es inicializado por un actor, el caso de uso siempre describe un servicio que debe dar el software a crear.
- **Diagramas de actividades:** Se pueden considerar una variedad tanto del diagrama de estados como los diagramas de interacción, ya que sirve para describir los estados de una actividad, que es un conjunto de acciones en secuencia y/o concurrentes en el cual intervienen clasificados
- **ERS**: Especificación de requisitos de software
- **Formato de alto nivel:** Descripción rápida de todos los procesos que realiza el diagrama de caso de uso
- **Formato expandido:** Muestra a detalle cada paso del diagrama de caso de uso, este formato se lo utiliza para obtener una mejor comprensión de los procesos y requisitos del sistema.
- **Herramienta CASE**: Herramienta que ayuda a modelar Lenguaje Estandarizado UML para solucionar problemas de la Ingeniería de software
- **Matriz de trazabilidad:** Contiene información entre los requerimientos del sistema para poder encontrar así su dependencia y los componentes del diseño.
- **Requisitos de usuarios:** Necesidades que los usuarios expresan verbalmente
- **Requisitos del sistema:** Son los componentes que el sistema debe tener para realizar determinadas tareas
- **Requisitos funcionales:** Servicios que el sistema debe proporcionar
- **Requisitos no funcionales:** Restricciones que afectaran al sistema
- **Software de modelado:** Técnica por la cual se debe tratar la complejidad inseparable de un sistema, el software de modelado ayuda a mostrar a los desarrolladores como se debe de tratar al sistema para su construcción
- **Unified Modeling Language:** Conocido como UML es un lenguaje estandarizado que le permite al desarrollador construir, modelar, diagramar y documentar todos las partes que contiene un software que este orientado a objetos o POO.

SOLUCIONARIO (EVALUACIÓN)

1. A) Unified Modeling Language.
2. B) Paradigma Orientado a Objetos.
3. D) Requisitos de usuarios, sistemas, funcionales y no funcionales.
4. C) Muestra la especificación de requisitos de software.
5. A) Entre los requisitos y los requisitos con los casos de uso.
6. C) EL documento ERS están definidas por l estándar IEEE 830-1998.
7. D) Se documenta con texto informal.
8. A) Identificar actores.
9. D) Parte Superior del Caso de Uso expandido.
10. C) Actor, Casos de Uso y relaciones

5

Modelo Conceptual.

Competencias

- Elabora modelos de los conceptos del sistema y del DER.

Cuando haya desarrollado esta unidad el **resultado de aprendizaje será**:

Aplicar el modelado conceptual y diagramas entidad relación.

Contenidos

5.1 Diagrama Conceptual.

Introducción al Capítulo V

Un modelo conceptual tiene como objetivo principal identificar y explicar los conceptos más relevantes en un dominio de problema, identificando los atributos y las asociaciones existentes entre ellos.

Es denominado, también como una representación de las cosas, entidades, ideas, conceptos u objetos del "mundo real" o dominio de interés. Es necesario diferenciar que dichas entidades u objetos no componen el software.

También podría ser considerado como un diccionario visual de abstracciones, conceptos, vocabulario e información del dominio de problema.

No es definitivamente correcta o incorrecta, su intención en ser útil sirviendo como una herramienta de comunicación.
Es importante saber que el modelo conceptual también puede ser llamado: modelo de objetos, del dominio o de análisis; en cambio la clase conceptual, también puede ser llamado como concepto o entidad.
Así mismo el modelo conceptual muestra una serie de elementos que permiten una vista más amplia de lo que se está realizando, ya que no solo se limita a un marco conceptual, también incluye roles, asociaciones, atributos, generalización, agregación y composición, multiplicidad.

5 Modelo Conceptual

"Los modelos presentan un marco conceptual donde reflejar las teorías, plasmar propiedades y establecer los principios del diseño de los sistemas. Su importancia radica en que permiten identificar, organizar y realizar razonamientos sobre los componentes y comportamiento del sistema, son la guía para el proceso de diseño del software y pueda usarse posteriormente como una referencia para evaluar un diseño particular, razonar sobre la solución realizada y sobre el posible espacio de soluciones. Por su naturaleza debe ser expresivos, fáciles de usar y completos." [50]
Un modelo conceptual es un recurso esencial para el diseño de sistemas de software, nos proporciona una amplia visión sobre el comportamiento de los sistemas, así mismo brinda un espacio

prudente de soluciones para determinar si la solución realizada en el proceso del diseño del sistema es correcta.
"Estos modelos conceptuales, en algunos casos, se pueden representar mediante una ontología de conceptos y relaciones que suceden en el sistema." [51]

5.1 Diagrama Conceptual

Un diagrama representa gráficamente a un conjunto de elementos, los cuales están o no relacionados entre sí. Estos diagramas son utilizados para poder visualizar un sistema desde diferentes puntos de vista.

5.1.1 Definición

"Es la representación de cosas del mundo real y no de componentes de software. En él no se definen operaciones (o métodos); en este se pueden mostrar los conceptos, los atributos de los conceptos (opcionalmente) y la relación o asociación entre ellos. Informalmente podríamos decir que un concepto es una idea, cosa u objeto. Para descubrirlos debemos analizar los sustantivos en las descripciones textuales del dominio del problema, es decir, de la descripción del sistema, de los requerimientos y de los Casos de Uso." [52]

5.1.2 ¿Qué nos muestra el Modelo Conceptual?

"**Conceptos.** Elemento lógico o físico que ayuda a entender el problema, es parte del lenguaje utilizado por el cliente y generalmente se nombra como sustantivo. Se representan con el símbolo de una clase. Ejemplo: Cliente, Póliza y Domicilio." [53]
"**Atributos.** Información que caracteriza al concepto en el mundo real. Se muestra en el segundo compartimiento de las clases. Ejemplo: Nombre, apellidos y edad del cliente." [53]
"**Rol.** El rol también puede aclarar la relación entre dos conceptos, indica el rol que juega un concepto con respecto a otro en una relación de asociación. Ejemplo: Planes Aplicables al cliente." [53]
"**Generalización.** En lugar de poner una asociación para armar la frase "es un tipo de" podemos poner una generalización. Aunque esto podría crear confusión en los lectores no técnicos, por lo que hay que asegurarse que el lector del modelo entienda perfectamente el significado de la notación. Ejemplo: El Plan Oro es un tipo de plan de seguro de vida, al igual que el plan tradicional." [53]

"**Agregación y composición.** Indican una relación donde uno de los conceptos es el contenedor del otro. Ejemplo: la póliza contiene una Lista Beneficiarios." [53]

5.1.3 Estrategias de Desarrollo

Para poder desarrollar un diagrama de modelo conceptual debemos de referirnos al diagrama de caso de uso y al diagrama de actividades; porque en dichos diagramas se listan las descripciones de los objetos y asociaciones

El modelo conceptual debe permitir y asegurar la representación formal de los fenómenos del sistema; es decir que se debe realizar una modelación de la interfaz del software.

El modelo conceptual debe:

- Definir los elementos que van a interactuar entre ellos; los elementos son las abstracciones del sistemas; son las clases que interactúan en el software a crear.
- Definir las relaciones que se darán en la interfaz del software.
- La diagramación de los elementos con sus relaciones en conjunto.

Otra estrategia más abierta desde el punto de vista del usuario para el desarrollo del modelo conceptual es que:

El sistema debe estar enfocado en la meta del proyecto.

Definir las clases que van a interactuar en el sistema de manera que estén circunscritos en el tiempo; pero estos deben estar especificando su estado y tiempo que deben alcanzar.

Realizar una lista de todos los atributos que están inmersos dentro de una clase definiendo su tipo y su nivel de privacidad

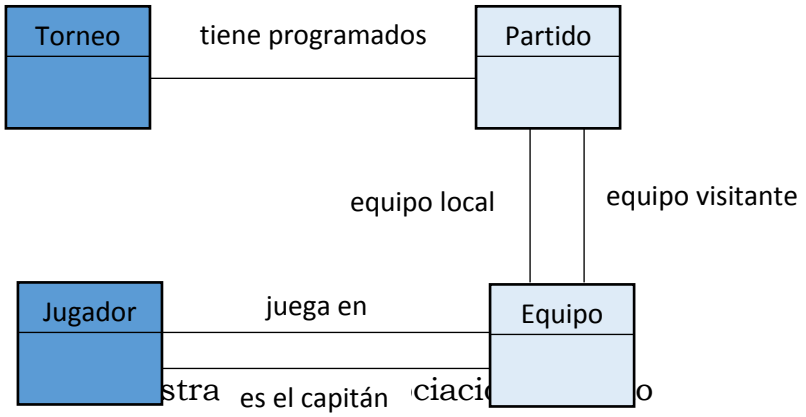
Realizar una lista de sus relaciones e intervenciones de los objetos (listar las relaciones de las actividades de cada clase).

5.1.4 Asociaciones

"Relaciones lógicas o físicas que existen en el mundo real entre dos conceptos. Si puedes armar una frase con dos conceptos significa que la puedes representar mediante una relación de asociación entre esos dos conceptos. Puedes colocarle el verbo que usas para relacionar los conceptos en la frase, indicándolo sobre la asociación con una punta de una flecha para indicar la dirección en que se debe leer la frase. Ejemplo: La Póliza cubre a un cliente asegurado, el cliente vive en un domicilio." [53]

Ejemplo 5.1. ¿Cómo mostrar que un equipo está conformado por varios jugadores, qué uno de ellos es el capitán, qué los equipos

juegan partidos o que los torneos tienen varios partidos programados?



5.1.5 Multiplicidad

"El número de instancias de un concepto relacionados con el otro concepto. Ejemplo: Una póliza tiene una lista de uno a diez beneficiarios." [53]

Es decir el número de instancias de tipo A que puedan asociarse o estén asociadas a un tipo B.

La multiplicidad tiene las siguientes expresiones:

- Exactamente uno
- 0..1 Cero o uno
- 0..* Cero o muchos
- 1..* Al menos uno: uno o muchos

Ejemplo 5.2. Continuando con el Ejemplo 5.1., ¿Cómo se modelan los siguientes aspectos?

- Un jugador puede jugar en sólo un equipo o en ninguno (si es que está libre).
- Todo equipo tiene un capitán.
- Los partidos los juegan dos equipos: uno será local y otro visitante.
- Todos los equipos tienen asegurados al menos un partido como visitante y al menos un partido como local.
- Cualquier torneo tiene al menos un partido programado.
- Todo partido se realiza en el contexto de un único torneo.

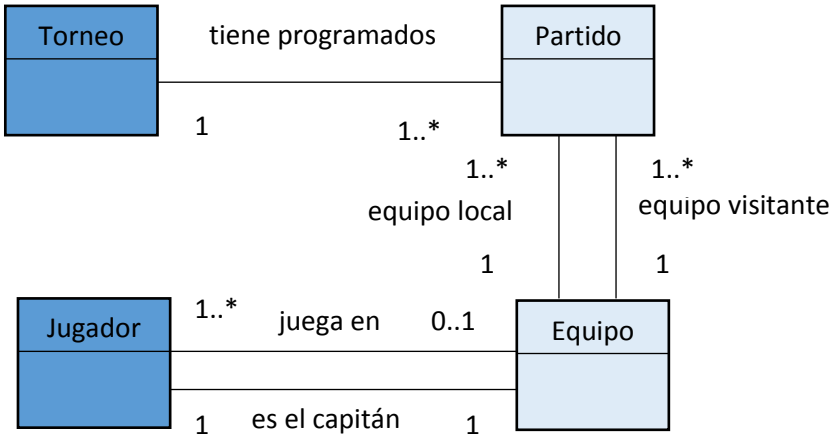


Ilustración 14: Multiplicidad ejemplo

5.1.6 Diagrama Conceptual

Ejemplo 5.3. Control de Proyectos – Diagrama Conceptual

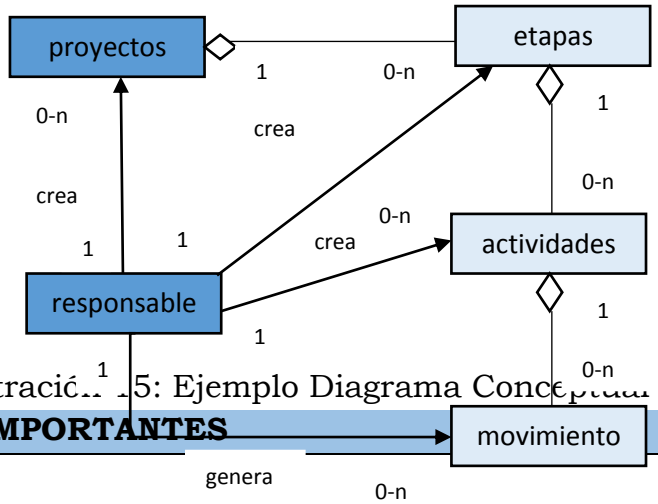


Ilustración 15: Ejemplo Diagrama Conceptual

CONCEPTOS IMPORTANTES

- **Asociaciones** permiten establecer relaciones entre dos conceptos que expresan alguna conexión significativa entre ellos.
- **Atributo** es la cualidad o característica del mundo real, que describe a un objeto o elemento.
- **Diagrama conceptual** es un gráfico que representa cosas reales y brinda una completa perspectiva de un sistema.
- **Multiplicidad** determina el número de instancias de un tipo que se puedan asociar a otro tipo, durante algún proceso.
- **Rol** es la representación de una entidad en una asociación.

LECTURAS COMPLEMETARIAS

Del articulo académico: "**Teorías y Modelos Conceptuales para un Diseño basado en Grupos**" de Garrido José Luis; Cañas José J., disponible en:
http://lsi.ugr.es/~mgea/invest/articulos/gea_interacc03.pdf

Desarrolle las siguientes preguntas:

1. ¿Qué es la teoría de la actividad?
2. ¿Qué pretende la cognición distributiva según Hutchins?
3. Elabore un gráfico sobre la modelización de sistemas complejos y explíquelo.
4. ¿Cuáles son los aspectos más relevantes del Marco Metodológico para Diseño de Sistemas Cooperativos (AMENITIES)?
5. Enumere y conceptualice los conceptos relacionados con la Estructura de AMENITIES.

Del articulo académico: "**Calidad en modelos conceptuales: un análisis multidimensional de modelos cuantitativos basados en la**

ISO 9126" de MARÍN BEATRIZ, CONDORI-FERNÁNDEZ NELLY, PASTOR OSCAR, disponible en: http://www.aemes.org/index.php/revista-de-procesos-y-metricas/numeros publicados/ano-2007-volumen-4/volumen-4-g-numero-especial-g-octubre-2007/179-calidad-de-modelos-conceptuales-analisis-multidimensional-de-modelos-cuantitativos-basados-en-la-iso-9126/download

Desarrolle las siguientes preguntas:

1. ¿Qué es la ISO 9126?
2. ¿Cuántos conjuntos propone el Modelo de calidad de Krogstie y cuáles son?
3. ¿Cuáles son los tipos de calidad que definen a los conjuntos del Modelo de calidad de Krogstie?
4. ¿Qué especifica el estándar ISO/IEC 14598?
5. Enumere las sub-características de la Funcionalidad y defina a dos de ellas.
6. ¿En que están enfocadas las Métricas de Li y Henry?

TALLER

Taller N°1 – Modelo Conceptual

1. **Escriba las estrategias para identificar asociaciones en un modelo conceptual.**

   _____
   _____
   _____
   _____
   _____
   _____
   _____
   _____

2. **Escriba las expresiones de la multiplicidad**

   _____
   _____
   _____
   _____

3. **En el mundo real un aeropuerto se considera concepto, ¿Por qué?**

_____

_____

Taller N°2 – Modelo Conceptual

1. **Obtener el modelo conceptual inicial para el caso de uso de negocio Registrar Pedido, a partir de las siguientes clases conceptuales (entidades):**

_____

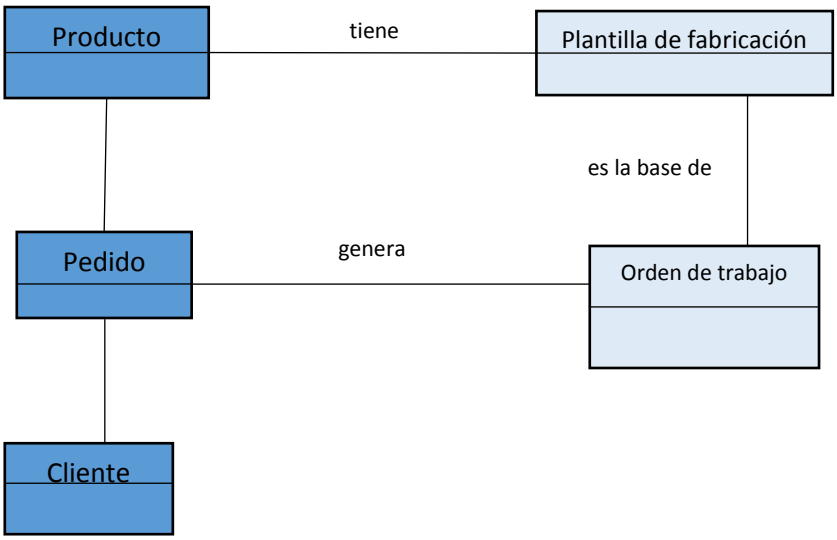
_____

_____

_____

_____

2. **Aplicar asociación y multiplicidad para el modelo.**



Taller N° 3 – Modelo Conceptual

1. **Elabore un cuadro comparativo entre los modelos conceptuales y modelos del sistema.**

Modelos conceptuales	Modelos del sistema
_____	_____
_____	_____
_____	_____

2. **Escriba los distintos modos para simplificar el estudio del objeto del modelo.**

_____
_____
_____

3. **Describa a los modelos conceptuales y modelos del sistema**

_____
_____

EVALUACIÓN		
1	**FORMATO**	**SIMPLE**
	Contexto	
	Planteamiento	**El modelo conceptual determina si el sistema desarrollado es correcto o incorrecto.**
	OPCIÓN A	Verdadero
	OPCIÓN B	Falso
	OPCIÓN C	---
	OPCIÓN D	---
	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios
2	**FORMATO**	SIMPLE
	Contexto	

	Planteamiento	**Un diagrama conceptual representa _____ a un conjunto de _____, los cuales están o no relacionados entre sí.**
	OPCIÓN A	gráficamente, elementos
	OPCIÓN B	conceptualmente, entidades
	OPCIÓN C	gráficamente, entidades
	OPCIÓN D	conceptualmente, elementos
	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios
3	**FORMATO**	SIMPLE
	Contexto	
	Planteamiento	**Es parte del lenguaje utilizado por el cliente y generalmente se nombra como sustantivo**
	OPCIÓN A	Rol
	OPCIÓN B	Entidad
	OPCIÓN C	Relación
	OPCIÓN D	Concepto
	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios
4	**FORMATO**	SIMPLE
	Contexto	
	Planteamiento	**Indican una relación donde uno de los conceptos es el contenedor del otro.**
	OPCIÓN A	Generalización
	OPCIÓN B	Asociación
	OPCIÓN C	Agregación y composición
	OPCIÓN D	Multiplicidad
	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios
5	**FORMATO**	SIMPLE
	Contexto	

	Planteamiento	**Puede aclarar la relación entre dos conceptos**
	OPCIÓN A	Rol
	OPCIÓN B	Entidad
	OPCIÓN C	Asociación
	OPCIÓN D	Concepto
	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios
6	**FORMATO**	SIMPLE
	Contexto	
	Planteamiento	**Para poder desarrollar un Diagrama de modelo conceptual debemos de referirnos a los diagramas de:**
	OPCIÓN A	Gantt
	OPCIÓN B	Secuencia
	OPCIÓN C	Casos de uso y actividades
	OPCIÓN D	Formato expandido
	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios
7	**FORMATO**	SIMPLE
	Contexto	
	Planteamiento	**El sistema debe estar enfocado en la meta del proyecto.**
	OPCIÓN A	Verdadero
	OPCIÓN B	Falso
	OPCIÓN C	---
	OPCIÓN D	---
	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios
8	**FORMATO**	SIMPLE
	Contexto	

	Planteamiento	**El ejemplo: La Póliza cubre a un cliente asegurado, el cliente vive en un domicilio, pertenece al concepto de:**
	OPCIÓN A	Multiplicidad
	OPCIÓN B	Asociación
	OPCIÓN C	Rol
	OPCIÓN D	Concepto
	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios
	FORMATO	SIMPLE
	Contexto	
	Planteamiento	**Que aspecto del modelo conceptual podría crear confusión en los lectores no técnicos**
	OPCIÓN A	Asociación
	OPCIÓN B	Rol
	OPCIÓN C	Agregación y composición
	OPCIÓN D	Generalización
	Respuesta Correcta	
	Nivel	1
9	**Operación Cognitiva**	Aplicación de Conceptos y Principios
	FORMATO	SIMPLE
	Contexto	
	Planteamiento	**Cuál de las siguientes expresiones pertenece a la multiplicidad**
	OPCIÓN A	0..1
	OPCIÓN B	1..n
	OPCIÓN C	*..0
	OPCIÓN D	1..0
	Respuesta Correcta	
	Nivel	1
10	**Operación Cognitiva**	Aplicación de Conceptos y Principios

GLOSARIO

- **Caso de uso:** es la descripción de pasos o actividades que deben realizarse para cumplir con un proceso.
- **Diagrama de actividades:** presenta o muestra un proceso de negocio o software como un flujo de trabajo, a través de una serie de acciones.
- **Diagrama:** es una representación gráfica de algún fenómeno con respecto a sus variaciones.
- **Instancia:** realización especifica de una clase.
- **Interfaz:** es la conexión funcional entre dos sistemas que brinda una comunicación en distintos niveles.
- **Modelo conceptual:** explica los conceptos significativos de un problema, además identifica los atributos y asociaciones existentes entre ellos.
- **Multiplicidad:** determina el número de instancias de un tipo que se puedan asociar a otro tipo, durante algún proceso.
- **Ontología:** definición formal de relaciones o asociaciones entre entidades que existen realmente para un dominio.
- **Requerimiento:** es una solicitud que se realiza con el fin de mejorar algún proceso.
- **Sistema:** conjunto de normas o actividades que se encargan de regular el funcionamiento de un grupo o colectividad.

SOLUCIONARIO (EVALUACIÓN)

1. B) Falso
2. C) gráficamente, elementos
3. D) Concepto
4. C) Agregación y composición
5. A) Rol
6. C) Casos de uso y actividades
7. A) Verdadero
8. B) Asociación
9. D) Generalización
10. A) 0..1

6

Comportamiento de los Sistemas

Competencias

- Elabora diagramas de secuencia y especifica contratos de operación.

Al desarrollar esta unidad el **resultado de aprendizaje será**:

- Aplicar diagramas para presentar los mensajes y las operaciones de los sistemas.

Contenidos

6.1 Diagramas de Secuencia.
6.2 Contratos de las Operaciones.

Introducción al Capítulo VI

Un sistema está formado por partes las cuales están relacionadas operativamente formando un solo objeto al cual nos interesa conocer su comportamiento en general.

Entonces para comprender el comportamiento del sistema se logra realizando los diagramas de secuencia del sistema que esté desarrollando y a su vez por cada operación realiza en cada diagrama realizar los contratos de operación.

Dentro de este capítulo se da a conocer una breve descripción de los comportamientos de los sistemas, primeramente se establecerán conceptos primordiales sobre los diagramas de secuencia y contratos de operaciones, los elementos que intervienen para poder realizarlos de la manera correcta, seguidamente se explicará los pasos a seguir para la elaboración de dichos diagramas y contratos.

6 Comportamiento de los Sistemas

Los comportamiento de los sistemas es aquel conjunto de elementos que determinan como el sistema va a responder a un evento externo provocado por un actor, este actor no necesariamente debe ser una persona también puede ser algún otro objeto del sistema o bien otro sistema que desea satisfacer alguna necesidad.

Para conocer los eventos que los actores pueden solicitar al sistema, se realizan diagramas de secuencia, el cual da conocer las relaciones que existe entre los distintos objetos de la aplicación a lo largo del tiempo, al igual que las operaciones y eventos que un actor externo realiza.

Diagramas de Secuencia

Los diagramas de secuencia, al igual que muchos otros diagramas utilizados durante todo el proceso de la ingeniería de software, son de mucha utilidad debido a que muestra las relaciones existentes entre los distintos objetos de una aplicación a lo largo del tiempo (de ahí su nombre “secuencial”). Cabe resaltar que estos diagramas se realizan en la etapa de diseño y se modelan a partir de los diagrama de caso de usos. Un caso de uso contiene una vista superficial de un escenario en cambio con la utilización de los diagramas de secuencia se logra dar más detalles sobre la implementación de un escenario, tales como objetos, clases y mensajes entre objetos.

6.1 Definición

"Los diagramas de secuencia muestra en forma gráfica los eventos que fluyen de los actores al sistema, es decir, buscan describir en forma particular el curso de los eventos en un caso de uso" [54].
"Son aquellos diagramas que muestran la secuencia de eventos entre los actores y el sistema el cual permiten identificar las operaciones del sistema" [55].
"Los diagramas de secuencia muestran el intercambio de mensajes (es decir la forma en que se invocan) en un momento dado. Los diagramas de secuencia ponen especial énfasis en el orden y el momento en que se envían los mensajes a los objetos" [56].
"Es uno de los diagramas que permiten modelar el comportamiento dinámico del sistema, permite definir cómo interactúan y colaboran los diferentes elementos del software que se tiene que desarrollar con el fin de llevar a cabo las funcionalidades requeridas" [57].
Dada las definiciones mencionadas anteriormente, se puede decir que los diagramas de secuencia muestran la interrelación existente entre varios objetos al igual que la secuencia o el orden en que va ocurriendo estas relaciones, en otras palabras especifican la comunicación entre objetos.

6.1.1 Actividades y Dependencias

Las actividades y dependencias son conceptos muy utilizados en la programación orientada a objetos, una dependencia en POO se define como "la relación que existe entre dos clases" [58], es decir que una clase va a depender de la acción que realice otra clase para cumplir con una tarea en específica.
La dependencia se representa con una flecha semi-cortada, como se muestra en la figura.

Ilustración 16: Instancia o Dependencia

En un diagrama de secuencia, las dependencias serán todos aquellos mensajes que un objeto desea conocer de otro.
Existen diferentes tipos de dependencia, la utilización de la misma va a depender de la metodología de desarrollo que utilice para la elaboración de su proyecto, estas dependencias son las siguientes:

- Obligatoria: como su palabra lo indica, es aquella dependencia que se debe cumplir de forma obligatoria.
- Discrecional: "determina una secuencia o relación en base al conocimiento del trabajo a realizar, pero podrá establecerse otra secuencia también aceptable. Este tipo de dependencia debe documentarse" [54].
- Externa: cuando una actividad depende del resultado de una actividad que esté externa al proyecto o en otro diagrama de secuencia.
- Interna: son las más comunes, en estas dependencias, las actividades no pueden iniciar hasta que finalice la actividad anterior.

6.1.2 Eventos y Operaciones

Los diagramas de secuencia fueron diseñados para mostrar los eventos producidos por parte de los actores externos hacia el sistema, esto se lo conoce como eventos de entrada, y el sistema al responderle al usuario genera un evento de salida.

Para establecer los eventos dentro de un diagrama de secuencia, se debe consultar previamente los casos de usos, que son los que muestran la forma en cómo interactúan los actores con el sistema, cada interacción que realice el actor va a generar nuevos eventos, y el actor para cumplir con cada evento utiliza los mensajes y operaciones, por ejemplo si un usuario desea consultar su saldo actual en un cajero automático, deberá ingresar su usuario y contraseña (evento de entrada), este evento de autenticación inicia una operación dentro del sistema.

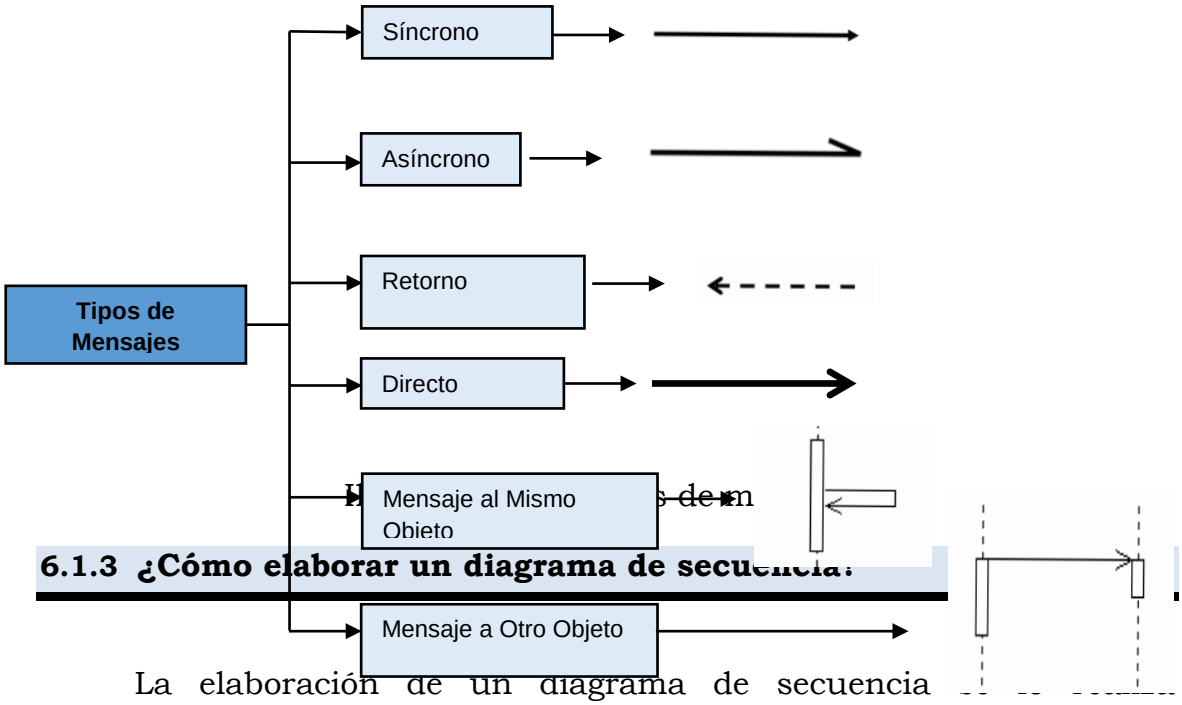
Las operaciones más conocidas como métodos o procedimientos, son muy utilizados por los mensajes en los diagramas de secuencia, pero para entender cómo funcionan las operaciones se debe conocer previamente que son los mensajes y operaciones.

Una operación se define como "un conjunto de instrucciones cuyo objetivo es realizar una tarea en específico". [59]

"Las operaciones de un objeto son consecuencia de un estímulo externo representado como mensaje enviado desde otro objeto" [59].

"Un mensajes es una comunicación entre objetos que transmite información con la expectativa de desatar una acción" [59].

Los mensajes se muestran como flechas, pueden ser síncronos o asíncronos. Un mensaje síncrono se lo caracteriza debido a que posee una saeta oscura en cambio los asíncronos por una saeta blanca. Si un mensaje tiene líneas continuas es un mensaje de llamada y si tiene líneas punteadas es un mensaje de retorno.



6.1.3 ¿Cómo elaborar un diagrama de secuencia?

La elaboración de un diagrama de secuencia siguiendo estos pasos:

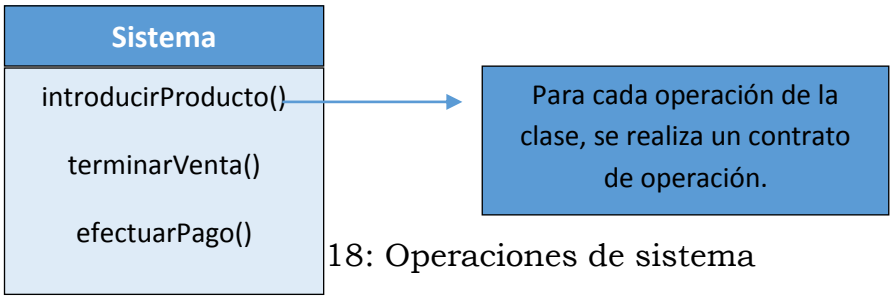
1. Tener a la mano los casos de uso, por cada caso de uso se realiza un diagrama de secuencia.
2. Identificar los objetos que intervienen en el caso de uso, luego se los coloca en la parte de arriba del diagrama, en el eje de las X.
3. Identificar los objetos que van a iniciar con la interacción, a estos objetos se lo coloca en la parte izquierda y los objetos sobrantes en la parte derecha.
4. Realizar las dependencias que tiene un objeto con otro, para esto se utiliza los mensajes, el objeto que envía y el que lo recibe va en el eje de las "Y", de arriba hacia abajo.
5. En cada mensaje se coloca la operación.
6. Cada objeto debe tener una línea de vida.
7. Y dependiendo de la duración del mensaje, se dibuja el foco de control de cada objeto.

6.2 Contratos de las Operaciones

Una vez realizado los diagramas de secuencia con todos sus mensajes e identificadas correctamente las operaciones, el siguiente paso es desarrollar los contratos de operaciones.

Los contratos de operaciones definen cual será el comportamiento del sistema al momento de realizar las operaciones planteadas en el diagrama de secuencia.

"Los contratos de operaciones ayudan a describir el comportamiento del sistema, describe el cambio de un estado a otro de los objetos del dominio como resultado de la ejecución de una operación" [60].
En los contratos de operaciones "se toma en cuenta el "qué" más que el "cómo", y se lo expresa en forma de pre y post condición en torno a cambios de estado" [60].
Se pueden elaborar contratos ya sea para algún método de alguna clase establecido en el diagrama de clases o bien para alguna operación global del sistema.



18: Operaciones de sistema

6.2.1 Elementos

El contrato de operación consta de los siguientes elementos:
Nombre de la Operación: Elemento que indica el nombre y que argumentos (opcional) contiene la operación.
Responsabilidades: Se escribe de manera informal, detallando cual será el propósito de la operación.
Pre-condiciones: Son los estados previos del sistema antes de realizar alguna operación.
Post-condiciones: Estado del sistema después de realizar alguna operación. Estas post condiciones pueden ser:

- Instancias de objetos
- Instancias de Asociaciones.
- Cambio de valor de un atributo.

Salida: Resultado de la operación.

6.2.2 Actividades y Dependencias de los Contratos

La elaboración de los contratos de operación se realiza en la fase de diseño y para proceder a crear los contratos de operaciones, previamente se debió haber realizado el modelo de dominio o conceptual, en conjunto con el diagrama de secuencia e identificando las operaciones que el sistema deberá realizar.

Estos contratos se realizan con el fin de definir cuál será el comportamiento del sistema, dando a conocer a través de las dependencias de los objetos, los efectos que tendrán sus operaciones en el sistema.

6.2.3 Secciones de un Contrato

Las secciones de un contrato se lo detallan en la siguiente tabla:
Contrato <Identificador> <Nombre> (aquí va el nombre del contrato)

Operación:	**Nombre de la operación y parámetros(opcional)**
Referencias Cruzadas:	Los casos de usos donde se encuentran las operaciones
Responsabilidades:	Las tareas que debe realizar la operación
Salidas:	Mensajes que envía el sistema.
Notas:	Alguna sugerencia como algoritmos, diagramas de flujo etc.
Excepciones:	En caso de que la operación produzca alguna excepción
Precondiciones:	Estado del sistema antes de que ocurra la operación.
PostCondiciones:	Estado del sistema después de que ocurre la operación

Tabla 9: Secciones de un contrato

6.2.4 Post condiciones - Precondiciones

Post-condiciones: Para elaborar las post-condiciones, se utiliza el modelo conceptual.
"Describen cambios en el objeto del modelo de dominio. Los cambios de estado del modelo de dominio comprenden la creación de instancias, formación o ruptura de asociaciones, y cambios de atributos" [61].
Las post-condiciones se dividen en tres categorías:

1. "Creación y eliminación de instancias" [61].
2. "Modificación de atributos" [61].

3. “Formación y ruptura de asociaciones” [61].

Cuando se escribe una post-condición se debe hacer utilizando el tiempo pasado, “para declarar sobre un posible cambio en el estado que ya ha pasado” [62].

Un ejemplo sería “se ha creado un usuario” en vez de decir “crear usuario”.

Al momento de instanciar un objeto, ese objeto está asociado a otro, por lo tanto al momento de escribir la post-condición, debe escribirse todas las asociaciones que ese objeto tuvo, olvidar nombrarlas es un error muy común al momento de escribir las post-condiciones.

Pre-condiciones: Para la elaboración de precondiciones se debe suponer cual será el posible estado del sistema al momento de que inicie la operación, se establece condiciones para que la operación realice su tarea exitosamente.

“Representan los requisitos que se deben cumplir antes de llamar a una operación, son condiciones que debe garantizar el programa que desee utilizar la operación” [63].

Las pre-condiciones que pueden declararse son:

- “Cosas que son importantes probar en el software en algún momento de la ejecución de la operación” [63].
- “Cosas que no serán sometidas a prueba pero de las cuales depende el éxito de la operación” [63].

CONCEPTOS IMPORTANTES

- **Actor.- “**Persona, unidad organizacional, mecanismo, u otro sistema que deseando satisfacer un requerimiento inicia la interacción con el sistema.” [61]
- **Contrato de operaciones.-** “Describen el comportamiento del sistema y el cambio de un estado a otro de los objetos del dominio como resultado de la ejecución de una operación” [60].
- **Diagramas de Secuencia.-** Muestran la interrelación existente entre varios objetos al igual que la secuencia o el orden en que va ocurriendo estas relaciones, en otras palabras especifican la comunicación entre objetos.
- **Eventos.-** Todas las acciones producidas por parte de los actores externos hacia el sistema.
- **Mensajes.-** Forma de comunicación que tiene los objetos que participan en un escenario para transmitirse información y dar como resultado una acción.

LECTURAS COMPLEMETARIAS

Del articulo académico: **"Extensión del Diagrama de Secuencias UML para el Modelado orientado a Aspectos"** de Cristian Vidal, disponible en: http://www.scielo.cl/pdf/infotec/v24n5/art02.pdf

Desarrolle las siguientes preguntas:

1. ¿Cuántas partes del software modela un diagrama de secuencia?
2. Mencione las partes principales del diagrama de secuencia.
3. Indique los 4 tipos de mensajes que soporta los diagramas de secuencia.
4. ¿Qué es el refinamiento en los diagramas de secuencia?
5. Nombre las metodologías que actualmente soporta el refinamiento.

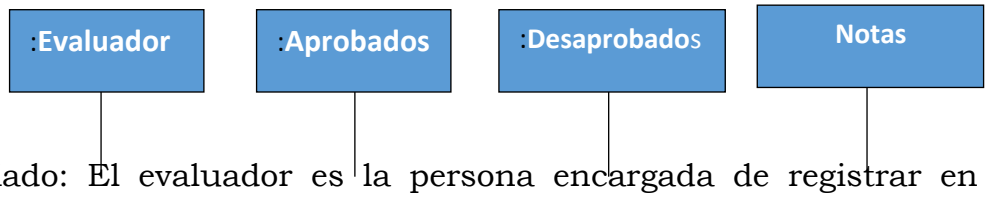
Del articulo académico: **"Generación del diagrama de secuencias de UML 2.1.1 desde esquemas pre-conceptuales "** de Carlos Mario Zapata, disponible en:
http://repository.eia.edu.co/revistas/index.php/reveia/article/view/212/208
Desarrolle las siguientes preguntas:

1. Escriba al menos dos usos de los diagramas de secuencia.
2. ¿Cómo define el autor al modelo verbal?
3. Escriba los elementos de los esquemas pre-conceptuales.
4. Mencione los pasos para la obtención de diagramas de secuencia a partir de esquemas pre-conceptuales.
5. Escriba la definición de reglas heurísticas.

TALLER

Taller N° 1 – Introducción a los Diagramas de Secuencia

En base al siguiente enunciado, escriba dos mensajes de los objetos que se encuentran en la siguiente gráfica:



Enunciado: El evaluador es la persona encargada de registrar en el sistema a los alumnos aprobados y desaprobados, cualquiera de los dos tipos de alumnos pueden consultar en el sistema sus notas, para consultar alguna nota debe previamente ingresar su usuario y contraseña, el evaluador puede modificar e ingresar las notas en el sistema.

Solución:

Taller N° 2 – Diagrama de Secuencia: Caso Práctico

Desarrollar un diagrama de secuencia con los siguientes datos:

Requerimiento: Retirar dinero de un AMT utilizando una tarjeta de banco.

Descripción: Un cliente solicita el retiro de dinero al AMT, el cajero (AMT) le pedirá que ingrese su tarjeta, el cliente ingresa la tarjeta y AMT lo verificará, luego consultará su saldo disponible y el cliente efectuará el retiro del dinero.

Solución:

Taller N° 3 – Contrato de Operaciones

Del diagrama de secuencia del taller 2, realice un contrato de operación para la operación verificarCuenta().

Solución:

Operación:	**verificarCuenta(user,pass)**
Referencias Cruzadas:	
Responsabilidades:	
Salidas:	
Notas:	
Excepciones:	
Precondiciones:	
PostCondiciones:	

EVALUACIÓN

1	**FORMATO**	**SIMPLE**
	Contexto	

	Planteamiento	**Indique la razón por la cual se conoce a los diagramas de secuencial como "secuencial".**
	OPCIÓN A	Por sus mensajes.
	OPCIÓN B	Las tareas se realizan en paralelo.
	OPCIÓN C	Muestra las relaciones existentes entre los distintos objetos de una aplicación a lo largo del tiempo
	OPCIÓN D	Contiene líneas de vida.
	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios
	FORMATO	SIMPLE
	Contexto	
	Planteamiento	**En qué etapa del ciclo de desarrollo de software se realizan los diagramas de secuencia.**
	OPCIÓN A	Diseño
	OPCIÓN B	Análisis
	OPCIÓN C	Codificación
	OPCIÓN D	Pruebas
	Respuesta Correcta	
	Nivel	1
2	**Operación Cognitiva**	Aplicación de Conceptos y Principios
	FORMATO	SIMPLE
	Contexto	
	Planteamiento	**Complete lo siguiente: Los diagramas de secuencia muestra en forma gráfica los -……- que fluyen de los -……- del sistema.**
	OPCIÓN A	Eventos-objetos
	OPCIÓN B	Estados- objetos
	OPCIÓN C	Objetos-eventos
	OPCIÓN D	Eventos-actores
	Respuesta Correcta	
3	**Nivel**	1

	Operación Cognitiva	Aplicación de Conceptos y Principios
4	**FORMATO**	SIMPLE
	Contexto	
	Planteamiento	**¿Los diagramas de secuencia permiten identificar las operaciones del sistema?**
	OPCIÓN A	Verdadero
	OPCIÓN B	Falso
	OPCIÓN C	
	OPCIÓN D	
	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios
5	**FORMATO**	SIMPLE
	Contexto	
	Planteamiento	**Los componentes de un diagrama de secuencia son:**
	OPCIÓN A	Objetos, vinculo, interacción
	OPCIÓN B	Objetos, foco de control, mensajes
	OPCIÓN C	Objetos vínculos.
	OPCIÓN D	Objetos, línea de vida, vinculo.
	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios
6	**FORMATO**	**SIMPLE**
	Contexto	
	Planteamiento	**Qué significa dependencia**
	OPCIÓN A	Cuando un objeto está contenido en otro.
	OPCIÓN B	Cuando un objeto depende del resultado de otro.
	OPCIÓN C	Instanciar un objeto.
	OPCIÓN D	Cuando un objeto hereda atributos y operaciones de la clase padre.

	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios
7	**FORMATO**	SIMPLE
	Contexto	
	Planteamiento	**Los elementos de un contrato son:**
	OPCIÓN A	Operación-referencias c. –responsabilidades – salidas- pre y post condición.
	OPCIÓN B	Actor – operación- responsabilidades
	OPCIÓN C	Línea de vida – pre y post condición – responsabilidades - salidas
	OPCIÓN D	Mensajes - Operación-referencias c. – responsabilidades – salidas- pre y post condición.
	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios
8	**FORMATO**	SIMPLE
	Contexto	
	Planteamiento	**¿Los contratos de operaciones ayudan a describir el comportamiento del sistema?**
	OPCIÓN A	Verdadero
	OPCIÓN B	Falso
	OPCIÓN C	
	OPCIÓN D	
	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios
9	**FORMATO**	SIMPLE
	Contexto	
	Planteamiento	**Las tareas que debe realizar la operación, se refiere al elemento de contrato de:**
	OPCIÓN A	Post condición

	OPCIÓN B	Pre condición
	OPCIÓN C	Salidas
	OPCIÓN D	Responsabilidad
	Respuesta Correcta	
	Nivel	1
	Operación Cognitiva	Aplicación de Conceptos y Principios
	FORMATO	SIMPLE
	Contexto	
	Planteamiento	**Estado del sistema antes de que ocurra la operación, se refiere al elemento de contrato de:**
	OPCIÓN A	Referencias cruzadas
	OPCIÓN B	Responsabilidades
	OPCIÓN C	Pre condición
	OPCIÓN D	Post Condición
	Respuesta Correcta	
	Nivel	1
10	**Operación Cognitiva**	Aplicación de Conceptos y Principios

GLOSARIO

- **Activación:** Se refiere a los cuadro de activación (foco) que se encuentran en la línea de vida de un objeto, estos cuadro representan cuanto tiempo se tardará el objeto en realizar una acción.
- **Comportamiento del sistema:** Manera en cómo responderá el sistema al realizarse algún evento externo por parte del usuario.
- **Destrucción de objetos:** Indica cuando un objeto deja de existir en la línea de vida, se lo presenta con una "X" o bien con la etiqueta <<destruir>>.
- **Línea de Vida:** Indica el tiempo de vida de un objeto.
- **Loop:** Forma de expresar una repetición (while, do-while, for) en un diagrama de secuencia y se lo representa con la figura del rectángulo.
- **Operación:** "Conjunto de instrucciones cuyo objetivo es realizar una tarea en específico" [59].

- **Post-condiciones:** Estado del sistema después de realizar alguna operación.
- **Pre-condiciones:** Estados previos del sistema antes de realizar alguna operación.
- **Responsabilidades:** Tareas que una operación debe realizar.
- **Rol de clase:** Nombre del objeto en un diagrama de secuencia, en él no se listan los atributos de la clase.

SOLUCIONARIO (EVALUACIÓN)

1. C) Muestra las relaciones existentes entre los distintos objetos de una aplicación a lo largo del tiempo.
2. A) Diseño
3. D) Eventos-Actores
4. A) Verdadero
5. B) Objetos, foco de control, mensajes
6. B) Cuando un objeto depende del resultado de otro.
7. A) Operación-referencias c. –responsabilidades – salidas- pre y post condición.
8. A) Verdadero
9. D) Responsabilidad
10. C) Pre-Condición

Bibliografía

[1] Real Academia Española, «Diccionario de la Real Academia Española,» 2014. [En línea]. Available: http://lema.rae.es/drae/srv/search?id=yHfxyKIWfDXX2kpDVRWA . [Último acceso: 03 01 2016].

[2] C. d. P. Heredero, Informática y comunicaciones en la empresa, Madrid: ESIC Editorial, 2004.

[3] V. F. Alarcón, Desarrollo de Sistemas de Informacion, Barcelona : Univ. Politèc. de Catalunya, 2006.

[4] R. C. Morales, Introducción Al Análisis de Sistemas Y la Ingeniería de Software, EUNED, 1993.

[5] W. T. H. Charles T. Horngren, Contabilidad, Quinta edicion, Monterrey : Pearson Educación, 2003.

[6] United Nations, Manual para la elaboración de un sistema de estadísticas sobre justicia penal, Nueva York: United Nations Publications, 2004.

[7] M. I. A. G. Sommerville, Ingeniería del software, Madrid : Pearson Educacion , 2005.

[8] J. E. Kendall, Análisis y diseño de sistemas, Mexico: Pearson Educacion, 2005.

[9] F. M. Lopez, Informáticos Generalitat Valenciana Grupos a Y B. Temario Bloque Específico Volumen i, Sevilla: MAD-Eduforma, 2005.

[10] J. R. L. F. D. R. Raúl Noriega Martínez, Curso de Ingeniería de Software, IT Campus Academy, 2015 .

[11] J. S. S. Garreta, Ingeniería de proyectos informáticos: actividades y procedimientos, Universitat Jaume , 2003.

[12] M. Delavaut, EDUCACION Y TECNOLOGIA: Un binomio excepcional, Grupo K.

[13] M. Torre Sampalo, E. Cortés y J. Prieto Tinoco, Informática, MAD, 2003.

[14] N. Rodríguez y W. Martínez, Planificación Y Evaluación de Proyectos Informáticos, Primera ed., San José: EUNED, 2006.

[15] I. Sommerville, Ingeniería del Software, Séptima ed., Madrid: PEARSON, 2005.

[16] Harvard business essentials, Gestión de proyectos: habilidades fundamentales para no salirse del presupuesto y cumplir los plazos, Barcelona: Grupo Planeta, 2004.

[17] J. Ramón Rodríguez, Gestión de proyectos informáticos: métodos, herramientas y casos, Barcelona: UOC, 2005.

[18] R. S. Pressman, Ingeniería de Software: Un enfoque práctico, Quinta ed., Madrid : McGraw Hill, 2002.

[19] F. J. Toro López, Gerencia de proyectos, ECOE, 2013.

[20] J. E. Kendall, Análisis y diseño de sistemas, Sexta ed., Pearson Educación, 2005.

[21] Á. Arias, Aprende sobre la Ingeniería del Software, IT Campus Academy, 2014.

[22] U. Medina Hernández y A. Correa Rodríguez, Cómo evaluar un proyecto empresarial: Una visión práctica, Madrid: Díaz de Santos, 2011.

[23] C. Calero Muñoz, M. G. Piattini Velthuis y M. Á. Moraga de la Rubia, Calidad del producto y proceso software, Ra-Ma, 2010.

[24] J. E. Kendall, Análisis y diseño de sistemas, Tercera ed., Pearson Educación, 1997.

[25] Universidad Nacional de Ingeniería, «Sistemas Uni,» [En línea]. Available: http://www.sistemasuni.edu.pe/dworkshops/ingenieria_requerimientos.html. [Último acceso: 04 Enero 2016].

[26] F. Alonzo, L. Martínez y Segovia Javier, INTRODUCCIÓN A LA INGENIERÍA DE SOFTWARE, Madrid : Grefol S.A , 2005.

[27] I. Sommerville, Ingenieria del software, Madrid : PEARSON EDUCACION , 2005.

[28] V. F. Alcarcón, Desarrollo de sistemas de información, Barcelona: Ediciones UPC , 2006.

[29] J. Valderrama, Información Tecnológica, Chile , 1997.

[30] D. B. Pena, Modelización de datos: Un enfoque practico, España: Lulu , 2009.

[31] B. C. Falgueras, Ingenieria del Software, Barcelona: UOC , 2003.

[32] R. Monferrer, «Especificación de Requisitos Software según el estándar de IEEE 830,» 2000.

[33] «fdi,» 22 Octubre 2008. [En línea]. Available: https://www.fdi.ucm.es/profesor/gmendez/docs/is0809/ieee830.pdf. [Último acceso: 05 Enero 2015].

[34] E. H. Meléndez, «GestioPolis,» 02 Mayo 2007. [En línea]. Available: http://www.gestiopolis.com/la-auditoria-interna/. [Último acceso: 05 Enero 2016].

[35] PMI, «Project Management Institute,» [En línea]. Available: http://americalatina.pmi.org/latam/pmbokguideandstandards/whatisastandar.aspx. [Último acceso: 05 Enero 2016].

[36] Synergix, «Tecnología y Synergix,» 7 Julio 2008. [En línea]. Available: https://synergix.wordpress.com/2008/07/07/requisito-funcional-y-no-funcional/. [Último acceso: 05 Enero 2016].

[37] M. d. C. G. Fuentes, «ConocimientosWeb.net,» 2011. [En línea]. Available: http://www.conocimientosweb.net/dcmt/ficha25180.html. [Último acceso: 04 Enero 2016].

[38] L. Alegsa, «Alegsa.com.ar,» 20 Marzo 2009. [En línea]. Available: http://www.alegsa.com.ar/Dic/requerimientos.php. [Último acceso: 05 Enero 2016].

[39] Guioteca , «Guioteca,» [En línea]. Available: http://www.guioteca.com/rse/que-son-los-stakeholders/. [Último acceso: 05 Enero 2016].

[40] J. Kendall, Análisis y diseño de sistemas., Pearson Educación, 2005.

[41] Sommerville & Alfonso Galipienso, Ingeniería del software, Madrid: Ingeniería del software., 2005.

[42] T. &. B. O. Weikiens, UML 2 Certification Guide: Fundamental & Intermediate Exams, Morgan Kaufmann, 2010.

[43] V. Fernández Alarcón, Desarrollo de sistemas de información: Una metodología basada en el modelado., Univ. Politèc. de Catalunya, 2006.

[44] S. Ceria, «IUn Método Práctico para Explorar Requerimientos,» UNIVERSIDAD DE BUENOS AIRES., 2008. [En línea]. Available: http://www-2.dc.uba.ar/materias/isoft1/2001_2/apuntes/CasosDeUso.pdf.

[45] L. A. A. F. &. S. P. F. J. Martínez Normad, Introducción a la ingeniería del software., Madrid: Delta Publicaciones, 2005.

[46] E. Mediavilla, Diagramas de casos de uso de UML: Instrucciones. Obtenido de Diagramas de casos de uso de UML, Pearson Educación,, 2008.

[47] V. F. Alarcon, Desarrollo de Sistemas de Información: Una metodologia basada en el modelado, Edicion UPC, 2006, p. 133.

[48] A. Arias, Aprenda sobre la Ingenieria de Software, Segunda ed., p. 108.

[49] B. C. Falgueras, Ingenieria de software, Primera ed., Barcelona: UOC, p. 99.

[50] M. G. José Luis Garrido, «Modelling dynamic group behaviours,» *Springer Verlag,* 2001.

[51] F. Paternò, Model-Based Design and Evaluation of Interactive Applications, Londres: Springer-Verlag, 1999.

[52] A. Canchala, «UML, ejemplo sencillo sobre Modelado de un Proyecto,» *Developer Network.*

[53] Abiztar Learning Technologies, «Dominando el Problema: Modelo Conceptual,» *Software Gurú,* 2012.

[54] D. H. Carmona, Teoría Genera de Sistemas. Un enfoque hacia la ingeniería de sistemas., Madrid: Pearson Educación S.A, 2011.

[55] E. T. López, Especificación de sistemas de software en UML, Barcelona: Universidad Politécnica de Cataluña, 2003.

[56] C. G. Cosío, Casos prácticos de UML, Madrid: Complutense, 2011.

[57] M. J. M. Galindo, Escaneando la informática, Barcelona: Sonia Poch, 2010.

[58] S. J. Valbuena, Programación orientada a Objetos, Armenia: Elizcom, 2011.

[59] M. Visconti, Fundamentos de Ingeniería de Software, España: Pearson Educación, 2011.

[60] C. Larman, Contrato de Operaciones del Sistema. UML y Patrones, Barcelona: Pearson Educacion, 2003.

[61] I. Sommerville, Ingenería del software, Madrid: Pearson Educación, 2005.

[62] D. J. Eck, La programación usando Java, Maracay: Creative Commons , 2009.

[63] F. A. Amo, Introducciòn a la Ingeniería del software, Zaragoza: Delta Publicaciones, 2005.

[64] J. M. P. Gómez, Diseño de Base de Datos Relacionales, Paraninfo. S.A., 2014, pp. 96-98.

[65] Vértice, Gestión de proyectos, Vértice, 2008.

[66] J. A. Ocaña, Gestión de proyectos con mapas mentales, vol. II, Club Universitario, 2013.

[67] L. Casal Otero, Gestión de proyectos: Elementos básicos a tener en cuenta como punto de partida para realizar eficazmente su

El programa de Reingeniería del Conocimiento en la Universidad Técnica de Machala (UTMACH) es un modelo emergente de gestión de la investigación que promueve saberes científicos con pertinencia social. Desde el Vicerrectorado Académico impulsamos la investigación colectivista, donde docentes y estudiantes se engranan en la construcción y divulgación del resultado de sus ejercicios pedagógicos, heurísticos y de vinculación social, en aras de contribuir con el fortalecimiento de nuestras ventajas comparativas y competitivas a nivel transfronterizo.

Mediante este programa estratégico la UTMACH impacta sus imaginarios respecto a la relación de la docencia con la investigación, muestra de ello es la presente obra donde se cristaliza el empoderamiento y profesionalismo de sus actores y redes al servicio de la formación crítica de profesionales de avanzada.

En la UTMACH seguimos conquistando el conocimiento a través de la investigación, por ello en cada acción emprendida ***proyectamos nuestra historia***.

Ing. Amarilis Borja Herrera, Mg. Sc.
VICERRECTORA ACADÉMICA.

REINGENIERÍA DE LA INVESTIGACIÓN

UTMACH