**LISTA DE COTEJO REPORTE DE INVESTIGACION VALOR: 30 %**

| | | |
|---|---|---|
| NOMBRE DEL DOCENTE: María de los Ángeles Pelayo Vaquero | | |
| DATOS GENERALES DEL PROCESO DE EVALUACIÓN | | |
| NOMBRE DEL ALUMNO:<br>PASCUAL MARTINEZ BRENDA JAZMIN | | |
| PRODUCTO:<br>INVESTIGACIÓN | UNIDAD: 2 | PERIODO ESCOLAR:<br>AGOSTO - DICIEMBRE 2024 |

| INDICADOR | VALOR | PORCENTAJE OBTENIDO |
|---|---|---|
| Presentación - Formato | 5 % | 5 % |
| Introducción<br>Idea clara del contenido del trabajo, motivando al lector a continuar con su lectura y revisión | 5 % | 5 % |
| Desarrolla el objetivo | 2 % | 2 % |
| Desarrollo de la investigación<br>La investigación cumple con el tema solicitado | 10 % | 10 % |
| Desarrolla la conclusión de investigación | 3 % | 3 % |
| Gramática y ortografía | 3 % | 3 % |
| Bibliografía | 2 % | 2 % |
| Total | **30 %** | **30 %** |

**LISTA DE COTEJO ENSAYO VALOR: 30 %**

| | | |
|---|---|---|
| NOMBRE DEL DOCENTE: María de los Ángeles Pelayo Vaquero | | |
| DATOS GENERALES DEL PROCESO DE EVALUACIÓN | | |
| NOMBRE DEL ALUMNO:<br>PASCUAL MARTINEZ BRENDA JAZMIN | | |
| PRODUCTO:<br>ENSAYO | UNIDAD: 2 | PERIODO ESCOLAR:<br>AGOSTO - DICIEMBRE 2024 |

| INDICADOR | VALOR | PORCENTAJE OBTENIDO |
|---|---|---|
| Presentación - Formato | 2 % | 2 % |
| Introducción<br>Idea clara del contenido del trabajo, motivando al lector a continuar con su lectura y revisión | 5 % | 5 % |
| Cuerpo del ensayo (empleo de artículos científicos) | 10% | 10% |
| Conclusión | 5 % | 5 % |
| Uso de formato de estilo | 2 % | 2 % |
| Gramática y ortografía | 3 % | 3 % |
| Bibliografía (APA o IEEE) | 3 % | 3 % |
| Total | **30 %** | **30 %** |

**EVALUACION 40%**

ANEXOS

# INSTITUTO TECNOLÓGICO SUPERIOR DE SAN ANDRÉS TUXTLA

Ensayo

**MATERIA:**

Fundamentos de telecomunicaciones

**ALUMNA:**

Brenda Jazmin Pascual Martínez

**DOCENTE:**

Maria De Los Ángeles Pelayo Vaquero

**CARRERA:**

ING. En informática

**GRUPO:**

310-A

**FECHA:**

04 de noviembre de 2024

# Introducción:

En el tema del desarrollo de software, las metodologías de desarrollo desempeñan un papel muy importante en la creación de aplicaciones que no solo cumplan con los requerimientos técnicos, sino que también se ajusten a las expectativas del usuario final, se entreguen a tiempo y con la calidad deseada. A lo largo de las últimas décadas, se han propuesto varias metodologías y enfoques para gestionar y optimizar este proceso, con el fin de abordar las diversas dificultades inherentes al desarrollo de software. Para ello se hablara de cuatro de las metodologías más relevantes: la creación del prototipo, la entrega incremental, el modelo en espiral de Boehm y el Proceso Unificado Racional (RUP). Cada uno de estos enfoques tiene sus particularidades, ventajas y limitaciones, y todos contribuyen a un entendimiento más profundo sobre cómo los equipos de desarrollo pueden gestionar el ciclo de vida del software de manera efectiva.

# Desarrollo:

## Creación del Prototipo

La creación de prototipos es una técnica de desarrollo de software que implica construir versiones preliminares del sistema (prototipos) con el fin de entender mejor los requisitos del cliente, obtener retroalimentación temprana y validar las expectativas. A diferencia del desarrollo tradicional que busca especificaciones completas antes de la implementación, la creación de prototipos promueve la construcción continua del sistema, permitiendo que los usuarios interactúen con las primeras versiones del software para ajustar y perfeccionar los requerimientos.

Una de las principales ventajas de este enfoque es la interacción constante entre los desarrolladores y los usuarios finales. Al poder ver y experimentar el software desde etapas tempranas del proyecto, los usuarios pueden proporcionar comentarios útiles que permiten ajustar la dirección del desarrollo. Sin embargo, una de las desventajas potenciales es que la creación de prototipos puede generar expectativas poco realistas, ya que los usuarios pueden asociar la primera versión del sistema con el producto final, cuando en realidad aún pueden existir importantes mejoras y modificaciones por hacer.

Este enfoque es especialmente útil cuando los requisitos no están completamente definidos al principio del proyecto, o cuando el cliente tiene dificultades para describir qué espera exactamente del producto final.

## Entrega incremental

El modelo de entrega incremental es una metodología que implica desarrollar el sistema en pequeñas partes o incrementos, entregando cada uno de estos incrementos al cliente de forma periódica. A medida que se completa cada incremento, se van agregando nuevas funcionalidades que forman parte del sistema final. Este enfoque permite que los clientes comiencen a usar el sistema desde las primeras fases del proyecto, y cada entrega posterior agrega más características o mejoras.

Una de las mayores ventajas que tienela entrega incremental es la reducción de riesgos. Al dividir el proyecto en pequeños segmentos, es más fácil gestionar el alcance, y los posibles problemas se pueden detectar de manera temprana. Además, los usuarios pueden comenzar a interactuar con el software desde el principio, lo que mejora su satisfacción y permite ajustes rápidos. Este modelo es muy adecuado para proyectos donde los requisitos son inciertos o están sujetos a cambios frecuentes, como es el caso de muchas aplicaciones web o sistemas móviles.

Sin embargo, la entrega incremental también puede presentar desafíos. La gestión del alcance del proyecto se vuelve crítica, ya que se pueden agregar nuevas funcionalidades a lo largo del proceso, lo que puede resultar en una ampliación no controlada del alcance o en funcionalidades que no son prioritarias para el cliente. Además, mantener la coherencia y la calidad en cada uno de los incrementos puede ser complicado.

## Modelo en Espiral de Boehm

El modelo en espiral, propuesto por Barry Boehm en 1986, es un enfoque de desarrollo que combina elementos del modelo en cascada y de los enfoques iterativos e increméntales. En este modelo, el proceso de desarrollo se organiza en una serie de ciclos o "espirales", cada uno de los cuales incluye cuatro fases principales: planificación, análisis de riesgos, ingeniería y evaluación del cliente. Este ciclo se repite varias veces, permitiendo la mejora continua del producto a medida que se avanza en el desarrollo.

La principal fortaleza del modelo en espiral es su enfoque en la gestión de riesgos. A través de cada iteración, los equipos de desarrollo evalúan los riesgos del proyecto y toman medidas para mitigarlos antes de avanzar al siguiente ciclo. Esto reduce la probabilidad de fallos costosos y mejora la calidad del software final. Además, el modelo es muy flexible y permite que los requisitos cambien y evolucionen durante el proceso, lo que lo convierte en una opción atractiva para proyectos complejos y de alto riesgo.

Sin embargo, una de las desventajas de este modelo es que puede ser difícil de gestionar, especialmente en proyectos grandes, debido a la necesidad de revisar constantemente los riesgos y adaptar el enfoque. Además, debido a su naturaleza iterativa, puede llevar más tiempo y ser más costoso que otros enfoques como el modelo en cascada.

El modelo en espiral es especialmente adecuado para proyectos grandes, complejos y con una alta incertidumbre en cuanto a los requisitos o la tecnología utilizada, como los desarrollos de sistemas críticos o proyectos de investigación y desarrollo.

## El Proceso Unificado Racional (RUP)

El Proceso Unificado Racional (RUP) es una metodología que se basa en un enfoque iterativo e incremental, que promueve la disciplina en cada fase del desarrollo del software. El RUP define un conjunto de buenas prácticas para la ingeniería de software, organizando el trabajo en cuatro fases: iniciación, elaboración, construcción y transición. En cada una de estas fases, se llevan a cabo iteraciones donde se desarrollan características específicas del sistema, asegurando que se construya una base sólida y escalable para el producto final.

Una de las principales ventajas del RUP es su enfoque en la gestión de riesgos y la calidad del software. A través de la iteración y la verificación continua, se identifican y resuelven problemas a medida que surgen, lo que reduce la posibilidad de fallos importantes. Además, RUP promueve una arquitectura basada en componentes, lo que facilita la escalabilidad y el mantenimiento del sistema a largo plazo.

Sin embargo, una de las críticas que se le hace al RUP es que puede ser percibido como pesado o burocrático, especialmente en proyectos más pequeños o con equipos menos experimentados. La necesidad de una planificación detallada y la implementación de numerosas prácticas puede hacer que el proceso sea más riguroso y menos flexible en comparación con otros enfoques ágiles.

RUP es adecuado para proyectos grandes y complejos donde la calidad del software y la capacidad de escalar el sistema son cruciales, como en el desarrollo de software empresarial o sistemas de misión crítica.

## Conclusión personal:

Las metodologías de desarrollo de software como la creación de prototipos, la entrega incremental, el modelo en espiral de Boehm y el Proceso Unificado Racional representan enfoques diversos que abordan las complejidades inherentes al proceso de creación de software desde distintas perspectivas. Cada una de estas metodologías ofrece ventajas significativas, como la capacidad de obtener retroalimentacion continua, mitigar riesgos, manejar proyectos complejos y garantizar la calidad del producto. Sin embargo, cada una también tiene limitaciones y es importante elegir detalladamente la más adecuada según el contexto del proyecto, el equipo de trabajo y las expectativas de los usuarios. En última instancia, la flexibilidad y adaptabilidad son clave fundamental para gestionar con éxito cualquier proyecto de desarrollo de software que se presente, y entender las fortalezas y debilidades de cada metodología es esencial para optimizar los resultados.

# INSTITUTO TECNOLÓGICO SUPERIOR DE SAN ANDRÉS TUXTLA

Investigacion

**MATERIA:**

Fundamentos de sistemas de información

**ALUMNA:**

Brenda Jazmin Pascual Martínez

**DOCENTE:**

Maria De Los Ángeles Pelayo Vaquero

**CARRERA:**

ING. En informática

**GRUPO:**

310-A

**FECHA:**

07 de noviembre de 2024

## Introducción:

Las herramientas CASE (Computer-Aided Software Engineering) son herramientas informáticas utilizadas en el desarrollo de software que ayudan a automatizar diversos procesos en las fases de análisis, diseño, implementación y mantenimiento de sistemas de software. Estas herramientas pueden incluir desde generadores de código hasta plataformas para la gestión de proyectos, pasando por entornos gráficos para el modelado de software y bases de datos.

Conforme ha transcurrido el tiempo, las herramientas CASE han progresado para cumplir con los requisitos de los programadores de software, posibilitando el aumento de la eficiencia, excelencia y la fusión de los variados elementos del proceso de desarrollo de vida. En el presente, existen herramientas CASE de pago (que necesitan una suscripción o compra) y herramientas CASE de acceso libre (de código abierto o gratuitas).

## Objetivo:

El propósito principal de este estudio es ofrecer una análisis detallado entre herramientas CASE comerciales y de código abierto, considerando los siguientes aspectos:

1. Explicación de las características principales de cada herramienta.
2. Evaluación de la facilidad de utilización y la curva de aprendizaje.
3. Evaluación de la adecuación con diversas plataformas de desarrollo.
4. Evaluación de la asistencia técnica y los documentos disponibles.
5. Examinar los gastos relacionados con cada variedad de instrumento.
6. Sugerencias sobre la alternativa más adecuada de acuerdo a los requerimientos del proyecto.

## Desarrollo:

A continuación, se presenta una tabla comparativa de algunas de las herramientas CASE más populares en ambas categorías

| *Herramienta* | *Tipo* | *Funcionalidades* | *Facilidad de uso* | *Compatibilidad* | *Costo* |
|---|---|---|---|---|---|
| **Enterprise Architect** | Comercial | Modelado UML, generación de código, análisis de requisitos, simulaciones. | Moderada | Windows, Linux, Mac | Licencia de pago |
| **Rational Rose** | Comercial | Modelado de sistemas, UML, análisis de requisitos, diseño de bases de datos. | Alta | Windows | Licencia de pago |
| **Visual Paradigm** | Comercial | Modelado UML, BPMN, generación de código, diseño de bases de datos. | Alta | Windows, Linux, Mac | Licencia de pago |
| **StarUML** | Libre (de código abierto) | Modelado UML, integración con otras herramientas de desarrollo. | Moderada | Windows, Linux, Mac | Gratuita (con funciones limitadas) |
| **Dia** | Libre (de código abierto) | Diagramas simples, diagramas UML básicos, diagramas de flujo. | Alta | Windows, Linux, Mac | Gratuita |
| **ArgoUML** | Libre (de código abierto) | Modelado UML, código en Java, integración con bases de datos. | Moderada | Windows, Linux, Mac | Gratuita |
| **PlantUML** | Libre (de código abierto) | Generación de diagramas UML a partir de texto. | Alta (para usuarios avanzados) | Windows, Linux, Mac | Gratuita |
| **Gliffy** | Comercial (freemium) | Creación de diagramas, soporte para colaboración en tiempo real. | Alta | Navegadores web | Plan gratuito y premium |
| **Lucidchart** | Comercial (freemium) | Diagramas UML, BPMN, flujo de trabajo, integraciones con Google Drive, etc. | Alta | Navegadores web | Plan gratuito y premium |

Las herramientas CASE se diseñan para facilitar y automatizar las actividades del ciclo de vida del software, principalmente en las fases de análisis y diseño. Permiten una mejor gestión de los proyectos, ya que ofrecen soporte para la planificación, programación, control de versiones, pruebas, y documentación.

**Las herramientas CASE pueden clasificarse en dos categorías principales:**

- **Herramientas CASE Comerciales:** Son soluciones desarrolladas por empresas especializadas, generalmente con un modelo de pago o suscripción. Suelen ofrecer características avanzadas, soporte técnico profesional y actualizaciones periódicas.
- **Herramientas CASE de Libre Distribución (Open Source):** Son gratuitas y de código abierto. Si bien algunas de ellas tienen limitaciones en términos de características avanzadas o soporte, son altamente flexibles y personalizables.

## Conclusión personal:

En mi punto de vista, la selección entre herramientas CASO comerciales y de distribución gratuita se basa principalmente en el tipo de proyecto y en la cantidad de dinero disponible.A pesar de su alto precio, las herramientas comerciales proporcionan un conjunto de funciones y asistencia que pueden validar el gasto en proyectos de gran escala. Además, las herramientas de código abierto son ideales para proyectos pequeños o equipos ágiles que requieran flexibilidad y control en su entorno de desarrollo, sin descuidar la calidad del proceso de ingeniería de software.

**Bibliografía:**

[1] Sommerville, I. (2011). Software Engineering (9th ed.). Addison-Wesley.

[2] Pressman, R. S. (2014). Software Engineering: A Practitioner's Approach (8th ed.). McGraw-Hill.

[3] ørgensen, M. (2013). Software Engineering: Theory and Practice. CRC Press.

# EVALUACION UNIDAD 2. FUNDAMENTOS DE SISTEMAS DE INFORMACION

30%

INGENIERIA INFORMATICA PERIODO SEPTIEMBRE - DICIEMBRE 2024

Se ha registrado el correo del encuestado (**231u0347@alumno.itssat.edu.mx**) al enviar este formulario.

NOMBRE Y APELLIDOS *

Brenda Jazmin Pascual Martinez

GRUPO: *

310-A

Definición de sistemas de información * 5 punt

Es un conjunto de componentes interrelacionados que recolectan (o recuperan), procesan, almacenan y distribuyen información para apoyar los procesos de toma de decisiones y de control en una organización.

Concepto de sistema de información * 5 punt

Es el conjunto formal de procesos que, al funcionar sobre una colección de datos organizados según las necesidades de una empresa, recopila, elabora y distribuye la información necesaria para su operación, así como para las actividades de dirección y control pertinentes.

¿Qué son datos? * 5 punt

Los datos son símbolos que no se eligen al azar y que reflejan valores de características o eventos. En este sentido, representan hechos, sucesos y transacciones que se han guardado en un formato acordado. Se obtienen a través de la lectura, la observación, el cálculo, la medición, entre otros métodos.

/1/2

Explicar las funciones de los sistemas de información * 10 puntos

Funciones de captación y recolección de datos: Esta función implica recoger información, tanto interna como externa, y enviarla a través del sistema de comunicación a los órganos encargados de organizarla, evitando duplicaciones e información irrelevante.
Funciones de almacenamiento: Consiste en agrupar los datos según algún criterio o en diferentes ubicaciones.
Tratamiento de la información: Su objetivo es transformar la información almacenada en datos útiles y significativos para quienes los necesiten.
Distribución o diseminación de la información: El sistema de información no solo debe ofrecer lo que cada usuario requiere, sino también difundir información a otros miembros de la empresa.

Explicar a clasificación de los sistemas de información * 10 puntos

Por nivel organizacional: Los sistemas pueden clasificarse en niveles operativo, administrativo, estratégico y de conocimiento.
Por modo de procesamiento de datos: Se pueden clasificar según el tipo de soporte que ofrecen.
Por designación funcional: Los sistemas pueden dividirse en sistemas de producción, financieros, comerciales, de marketing, entre otros.
Por objetos de control: Se pueden categorizar en sistemas de gestión empresarial, gestión de procesos tecnológicos, diseño asistido por ordenador, etc.
Por carácter del uso de la información de resultados: Se clasifican en gestores de información, asesores de información y sistemas de búsqueda de información.

Explicar las dimensiones de los sistemas de información. * 5 puntos

1/2

Funciones de captación y recolección de datos: Esta función se encarga de recolectar información, tanto externa como interna, y enviarla a través del sistema de comunicación a los responsables del sistema de información, quienes se encargan de organizarla para evitar duplicaciones y datos irrelevantes. Funciones de almacenamiento: Consiste en agrupar los datos según ciertos criterios o en diferentes ubicaciones. Tratamiento de la información: Su objetivo es convertir la información almacenada en datos útiles y significativos para quienes los necesiten. Distribución o diseminación de la información: El sistema de información no solo debe ofrecer la información requerida por cada usuario, sino también compartirla con otros miembros de la empresa.