# Resumen ingeniería de Software – Len

Los **Software son abstractos, intangibles y no tienen restricciones por los materiales ni leyes físicas, siendo elementos de un sistema lógico.** Entonces sumándolo con la ingeniería, no tiene limites naturales en la creación. **Mientras mas tiempo pasa, más complejos, difíciles de entender y costosos se hacen**

**La ingeniería de software busca crear un software a bajo costo, rápido, de buena calidad y mantenerlo a lo largo del tiempo**

**La calidad** puede ser:

* Que tan **mantenible** es
* Estabilidad
* **Velocidad**
* **Usabilidad**
* **Rendimiento**
* Legibilidad
* Tamaño
* **Costo**
* **Seguridad/Confiabilidad**
* Cantidad de fallos
* **Funcionalidad**
* Etc.

Y en **la disciplina de la IS** (Ingeniería de Software) abarca:

* **Diseño de software**
* Mejores practicas
* Tamaños óptimos
* **Proceso**
* **Entregar el software a tiempo**
* Cultura del lugar del trabajo
* **Prácticas de contratación**
* Etc.

El **software está en todos lados en la actualidad** **y de distinta variedad**. Esto hace que **sea complicado tener** **métodos universales** por ser tan variado. Por eso **cada proyecto es diferente (Puede fallar o salir bien)**

Factores de IS:

* **Demandas crecientes**: Cada vez se necesitan sistemas más grandes que requieren una mejor satisfacción de las demandas
* **Expectativas bajas**: Empresas chicas ignoran las técnicas de la ingería y luego la van aplicando poco a poco

## Historia, orígenes de la producción de Software como ingeniería

**“Ingeniería de software” se propuso en 1968 luego de crisis del software;**

**Dificultad de escribir software sin defectos, comprensibles y verificables. Bases inestables, complejidad, no había RRHH y falla de manejo de riesgos**. En 1979 y 1980 crearon las técnicas que se usan actualmente

## Actividades de Ingeniería de Software (IS):

* **Apoya desarrollo de software** profesional en lugar de la individual
* Técnicas de **especificación, diseño, evolución** de programas
* Se interesa en los aspectos de producción del software **(Especificaciones hasta mantenimiento)**
* Busca **resultados de calidad** **dentro de las fechas y presupuesto**
* **Sistemático y organizado**
* Selecciona **el método más adecuado** (A veces hay que ser creativos)

**Importancia del IS:**

* Cada vez se apoya mas en **sistemas de software**, entonces **necesitan ser confiables**
* A **largo plazo** es más **barato** **usar** métodos y técnicas de **IS**

**Proceso:**

* Proceso de software es sistemático (Secuencia de actividades)
* **Especificación, desarrollo, validación y evolución** del software
* Se necesitan **distintos procesos** dependiendo el tipo de software
* Sus actividades principales incluyen: **Comunicación, planeación, modelado, construcción, despliegue**

## Diseño de IS:

Representa la **arquitectura del sistema en forma de componentes**, **modelando interfaces**

* Área técnica de IS
* Viene **después de especificar requerimientos y pautas** para construir
* **Agrupa principios, conceptos y prácticas** en el desarrollo
* Es la mas deseada de hacer, por ser creativa
* Se **crea un modelo** con detalles de la arquitectura, datos y componentes
* **Modela el sistema o producto** y se usa para evaluar calidad y mejoras antes de hacer código

Después estos elementos son **analizados por si hay errores, inconsistencias u omisiones, buscando mejoras y viendo si es viable**

**Proceso de diseño:**

* Es un **proceso iterativo** donde se analizan los **requerimientos**
* Empieza siendo **abstracto** y se busca ser **lo más especifico posible**

Un **buen diseño** tiene:

* **Arquitectura con estilos y patrones**, hecha con **componentes**, de forma **evolutiva**
* **Distintas representaciones** de datos, interfaz y componentes
* **La estructura mas apropiada** para las clases
* **Componentes con características funcionales**
* **Interfaces que reduzcan complejidad de conexiones**
* Ser **eficaz**

**Tareas para hacer un diseño:**

1. **Estudiar modelo** de dominio y **diseñar estructura apropiada**
2. **Seleccionar arquitectura** adecuada
3. **Dividir subsistemas** de diseño
4. **Crear conjunto de clases** de diseño/componentes
5. **Diseñar** la interfazy las del **usuario**
6. **Diseñar** en el nivel de **componentes**
7. **Desarrollar** un modelo de despliegue

## Procesos de Software (Modelos de procesos)

Un proceso de software es **un conjunto de actividades que conducen a una creación** de un producto. Los procesos de Software son complejos, no existe uno ideal, cada uno hace su enfoque

**Modelos de Procesos:**

Un modelo es una **simplificación de la realidad** y tienen una perspectiva dependiendo cada uno

* **Descriptivos**: Informativos o analíticos mientras o post terminado el proceso realizado
* **Prescriptivos**: Definen actividades, acciones, tareas, hitos y artefactos (entregables) para un desarrollo de calidad. Generalmente son **Lineales, Incrementales y Evolutivos**

Modelo Cascada – 1970 Royce

* Modelo **Prescriptivo**
* Etapas **lineales**, no es recomendable regresar a la etapa anterior
* Cada etapa es autónoma y producen una salida o documento aprobado
* La siguiente fase comienza cuando termina la anterior
* **Ventajas**: Fácil de administrar, cada etapa devuelve artefacto (Documento, modelo, software). Cada fase bien definida, proceso claro por cliente
* **Desventajas**: Inflexible, visión estática, congela requisitos (Puede que al final no sea lo que el usuario necesita). Los clientes no participan en etapas posteriores a los requisitos, errores y omisiones de requisitos se ven al final del proyecto

Diagrama

Descripción generada automáticamente

***Nota:*** *Análisis de Requisito se cambio a Ingeniería de Requisito*

Modelo Incremental – 1980 – Mills

* Modelo **Prescriptivo**
* La **entrega y desarrollo está dividido en piezas**, haciendo que el usuario no espere hasta el final. Incrementando en cada una de las entregas
* Los requerimientos se priorizan, pudiendo agregar los prioritarios en etapas tempranas
* Artefactos de un incremento temprano como prototipo
* El usuario se **involucra**
* **Riesgos largos y complejos**
* Los errores en requisitos se detectan tarde
* Se entrega el software en “Partes pequeñas” llamadas “incrementos”
* **Ventajas:** Menos factor de riesgo, es más fácil detectar requerimientos para el siguiente incremento, si hay error se descarta solo la ultima iteración, reduce el tiempo de desarrollo

Diagrama

Descripción generada automáticamente

Modelo Prototipado 1984 – Floyd

* Modelo **prescriptivo**
* Construir mecanismos de software evaluados por el cliente en conjunto con el programador
* Es un proceso dentro de la construcción del software
* Hay 3 enfoques
  + **Exploratorio**: Etapa temprana, clasifica y elecita requisitos
  + **Experimental:** Simula aspectos desconocidos
  + **Evolutivo: Existen 2**
    - **Incremental:** Mejora gradual, no hay cambios de diseño, las necesidades se saben desde el principio
    - **Evolutivo:** Ciclos (Rediseño, reimplementación, reevaluación). Ambiente dinámico, los requisitos se descubren constantemente

**Forma, Flecha

Descripción generada automáticamente**

Modelo Espiral – 1988 – Boehm

* Modelo **prescriptivo**
* Cada ciclo de la espiral inicia identificando objetos o productos a elaborar y sus restricciones
* Cada espiral produce un producto que es una versión o componente de este
* Se basa en repetir todos los pasos por cada producto realizado
* Puede tener espirales en paralelo para desarrollar
* Modelo dirigido a los riesgos con costos y tiempos
* Analizar riesgos implica evaluar y prever caminos alternativos si es necesario
* **Ventajas:** Tiene ventajas de otros ciclos de vida, los prototipos iterativos son controlados. Puede adaptarse a lo largo del software, esto hace que se reaccione ante riesgos en cada nivel. Permite aplicar enfoque de construcción de prototipos en cualquier etapa de evolución. Es rápido
* **Desventajas:** Solo para proyectos de gran tamaño, carga de trabajo adicional, buena habilidad de resolución de riesgos, es complicado de hacer y muy complejo. No es muy usado

Diagrama, Dibujo de ingeniería

Descripción generada automáticamente

## Proceso Unificado - RUP

Propuesto por **IBM**, planea estratégicamente y administra riesgos. **Guían de forma natural del proceso de desarrollo de software** por lo que es considerado **estándar** para empresas

**RUP**

Es un **proceso** que permite **desarrollo de software a gran escala**, mediante **proceso continuos de pruebas y retroalimentación**, dando calidad. Eso si, con mayor esfuerzo administrativo, pero teniendo buenos beneficios

Es un **derivado del trabajo sobre UML,** siendo un **modelo hibrido** porque junta elementos de los modelos de proceso:

* Ilustra buenas prácticas y diseño
* Apoya creación de prototipos
* Entrega incremental

RUP tiene 3 perspectivas

* **Dinámica**
* **Estática**
* **Practica**

Fases:

**Construcción:**

* **Diseño, programación y pruebas**
* Se debe tener un software funcionando
* **Documentación relacionada**
* **Entregable para usuario**

**Transición:**

* Implementación del sistema
* **Es ignorada regularmente**, es una actividad costosa y problemática
* Esta fase termina con un software documentado que funcione en su entorno operativo

Cada fase es iterativa, con resultados incrementales

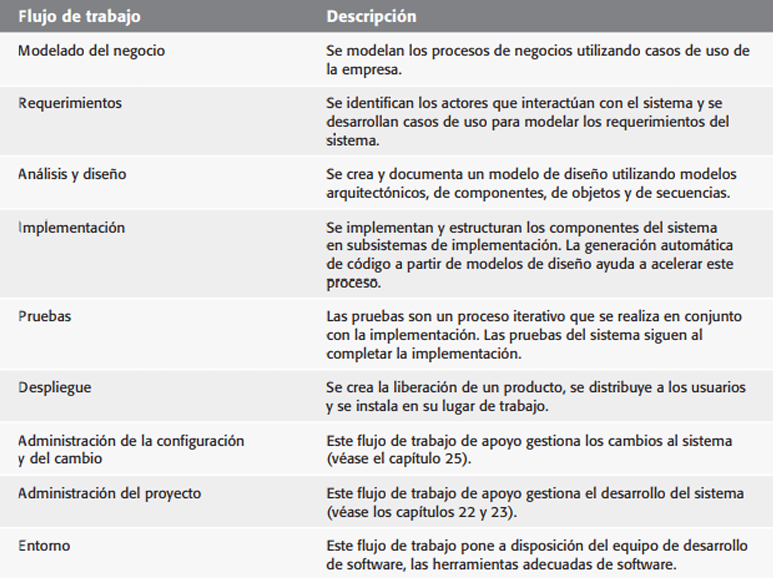
Gráfico

Descripción generada automáticamente

Las **actividades** de RUP son: (Posible que tome en examen esto)

* **Modelado del negocio**
* **Ingeniería de Requisitos**
* **Análisis y diseño, codificación**
* **Integración**
* **Pruebas**
* **Despliegue**
* **Actividades Sombrilla**

Iniciales (**M.I.A.I.P.D.A**)



RUP no es un proceso adecuado para todos los tipos de desarrollo, pero es potencial. Las innovaciones mas importantes son la **separación de fases y flujos.** **Siendo fases dinámicas y tiene metas**

**Flujos son estáticos y son actividades técnicas**, que se usan a lo largo del desarrollo

## Metodologías Agiles

**Reporte del Caos**: 1994, tasa de proyectos software era mala, se creó la metodología agiles

Es una **técnica de gestión**, se basa en **valores, principios, cambios de cultura**

Se enfoca en la creación constante de **valor agregado** en un entorno incierto a partir de **aprendizaje colaborativo** entre personas

Cumplir **manifiesto ágil**

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

El modelo Ágil hace que sea **flexible, de entrega rápida y mejora continua**

Diagrama

Descripción generada automáticamente

1. Nuestra mayor prioridad es **satisfacer** al cliente mediante la entrega temprana y continua de software con valor.
2. **Aceptar los cambios en los requisitos**, incluso al final del desarrollo. Los procesos ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
3. **Entregar** software funcional **frecuentemente**, con una preferencia por el **menor plazo** posible.
4. Los responsables de negocio y los desarrolladores deben **trabajar juntos** de forma cotidiana durante todo el proyecto.
5. Construir proyectos alrededor de individuos **motivados**. Darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.
6. El método más eficiente y efectivo de **comunicar** información dentro de un equipo de desarrollo es la conversación cara a cara.
7. El software **funcionando** es la principal medida de progreso.
8. Los procesos ágiles promueven el desarrollo sostenible. Los patrocinadores, desarrolladores y usuarios deben ser capaces de mantener un ritmo **constante** de forma indefinida.
9. La **atención** **continua** a la excelencia técnica y al buen diseño mejora la agilidad.
10. La **simplicidad**, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.
11. Las mejores arquitecturas, requisitos y diseños emergen de equipos **autoorganizados**.
12. **A intervalos regulares**, el equipo reflexiona sobre cómo ser más efectivo, para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

Ventaja de la agilidad

* Mayor satisfacción del cliente
* Mas motivado e involucrado
* Valor agregado sostenible al negocio
* Mejor calidad
* Detección temprana de errores
* Adaptable al cambio
* Fomentar la comunicación e interacción cara a cara

## SCRUM

Marco de trabajo de practicas emergentes de dominio complejos

Tiene

* 4 **roles**:
  + **Product Owner**: La voz del cliente, define prioridades, dueño del backlog
  + **Dev Team:** Desarrolla el producto, organiza trabajo, reporta progreso, multidiciplinado
  + **Stakeholder:** Observa y aconseja
  + **Scrum master:** Cuida el proceso, protege al equipo, refuerza reglas
* 3 **artefactos**:
  + **Product Backlog:** Lista de requerimientos, evoluciona, cualquiera agrega requerimientos
  + **Spring Backlog:** Lista de requerimientos del Sprint, administrada por Dev Team, siempre visible
  + **Incremento Producto:** Version del producto, potencial entregable, cumple definición de hecho
* 4 **eventos**:
  + **Sprint Planning:** 8 hrs, presentan requerimientos, estiman esfuerzos, se hace Spring backlog
  + **Daily meeting:** 15 minutos, que hice, que hago y que hare. Se actualiza Spring backlog
  + **Spring review/Demo:** 2 a 4 hrs, participan todos, informar y feedback
  + **Spring retrospective:**1 a 2 hrs, revisión de trabajo, identifica oportunidad de mejorar

El **Minimum Viable Product (MVP)** es la menor versión del producto con las características mínimas y necesarias para poder lanzarse al mercado

Ayuda a evitar construir cosas innecesarias y maximizar valor invertido

Dibujo de un gato con la boca abierta

Descripción generada automáticamente con confianza baja

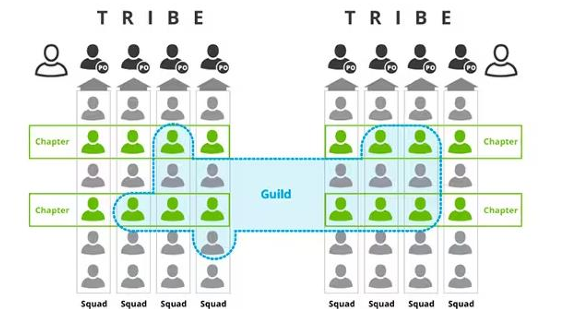
Escala de tiempo

Descripción generada automáticamente

## Modelo Spotify

Es una adaptación ágil centrada en escalar equipos ágiles a través de **tribus, escuadrones, capítulos y gremios, fomentando la autonomía y la alineación dentro de la organización​​**

Esta síntesis cubre los aspectos principales del documento relacionado con la ingeniería de software y su proceso de desarrollo, proporcionando una visión completa de los temas solicitados



Estructura Organizacional

**Escuadrones (Squads): Elijen herramientas, métodos y trabajo**

* **Descripción**: Equipos pequeños y autónomos de 6-12 personas.
* **Función**: Cada escuadrón se enfoca en una parte específica del producto, desde el diseño hasta el despliegue.
* **Autonomía**: Tienen la libertad de elegir sus herramientas y métodos de trabajo, fomentando la innovación y la responsabilidad.

**Tribus (Tribes): Comunica escuadrones, apoyan y guian**

* **Descripción**: Un grupo de escuadrones que trabajan en áreas relacionadas.
* **Función**: Facilitan la coordinación y la comunicación entre escuadrones que están interconectados.
* **Liderazgo**: Cada tribu tiene un líder de tribu que apoya y guía a los escuadrones sin interferir en su autonomía.

**Capítulos (Chapters): Dan conocimientos, técnicas, orientan y hacen reuniones**

* **Descripción**: Grupos de personas con habilidades similares dentro de una tribu.
* **Función**: Aseguran la coherencia técnica y comparten conocimientos entre escuadrones.
* **Liderazgo**: Un líder de capítulo (Chapter Lead) proporciona orientación técnica y organiza reuniones regulares.

**Gremios (Guilds): Comparten practicas/conocimientos entre tribus y escuadrones**

* **Descripción**: Comunidades de interés que cruzan tribus y escuadrones.
* **Función**: Permiten compartir prácticas y conocimientos en toda la organización sobre temas específicos, como DevOps, calidad, o diseño.

2. Principios y Cultura

**Autonomía y Alineación:**

* **Descripción**: Los equipos tienen la libertad de tomar decisiones (autonomía) pero están alineados con los objetivos de la empresa.
* **Función**: Este equilibrio asegura que los equipos innoven mientras trabajan hacia metas comunes.

**Transparencia:**

* **Descripción**: Se fomenta una cultura de apertura y transparencia.
* **Función**: Permite a los equipos aprender de los éxitos y fracasos de los demás, facilitando la mejora continua.

**Fail Fast:**

* **Descripción**: Promueve la experimentación rápida y el aprendizaje de los errores.
* **Función**: Minimiza el riesgo al permitir que los equipos prueben ideas nuevas sin temor al fracaso.

3. Ceremonias y Herramientas

**Ceremonias Ágiles:**

* **Scrum** **of** **Scrums**: Reuniones para coordinar entre escuadrones dentro de una tribu.
* **Reuniones** **de** **Capítulo**: Encuentros regulares para compartir conocimientos técnicos y resolver problemas comunes.

**Herramientas de Colaboración:**

* **Tableros** **Kanban**: Utilizados para visualizar el flujo de trabajo y gestionar las tareas.
* **Plataformas** **de** **Comunicación**: Herramientas como Slack o Microsoft Teams para la comunicación diaria.

Beneficios del Método Spotify

* **Escalabilidad**: Facilita la expansión de equipos sin perder agilidad.
* **Innovación**: Fomenta la creatividad y la experimentación.
* **Adaptabilidad**: Permite a los equipos adaptarse rápidamente a cambios en el entorno o en los requisitos del proyecto.
* **Cohesión**: Mantiene la alineación y la coherencia técnica a través de capítulos y gremios.

El Método Spotify es una evolución de prácticas ágiles, diseñado para maximizar la eficiencia y la satisfacción del equipo en entornos de desarrollo rápido y dinámico.

# Adicional

Diferencias entre las Metodologías Scrum y Spotify.

**Estructura Organizacional**: **Scrum** utiliza equipos **multifuncionales** llamados "Scrum Teams", mientras que **Spotify** utiliza "**Escuadrones**" dentro de "Tribus", "Capítulos" y "Gremios".

**Escalabilidad**: **Scrum** **se enfoca en equipos pequeños** y su coordinación a través de **"Scrum of Scrums**", mientras que **Spotify** tiene **un enfoque más detallado para escalar equipos** mediante su estructura de **tribus y capítulos.**

**Autonomía**: En **Spotify**, los **escuadrones tienen mayor autonomía** para elegir sus herramientas y métodos, mientras que, en **Scrum**, **hay más estructura y roles definidos** como Scrum Master y Product Owner.

**Alineación**: **Spotify** **enfatiza** **mucho la alineación entre escuadrones y tribus** para mantener una visión común, mientras que **Scrum se centra en la colaboración y comunicación dentro del equipo individual**.

### Posible para examen

**TEMA 1**

1. Sobre la siguiente frase: “En el contexto de la ingeniería de software, un proceso es una prescripción rígida que garantiza elaborar software de calidad”. Diga si es Verdadero o Falso y Justifique

2. Explique la diferencia entre actividades estructurales y actividades sombrilla. De un ejemplo de cada una.

3. ¿Cuáles son las actividades de un proceso de desarrollo RUP según Roger Pressman?

1. Modelado del negocio, Ingeniería de Requisitos, Análisis y diseño, codificación, Integración, Pruebas, Despliegue, Actividades Sombrilla.

2. Requerimientos, Análisis y diseño, codificación, Integración, Pruebas, Despliegue, Mantenimiento

3. Análisis y diseño, codificación, Integración, Pruebas, Despliegue.

4. Todas son correctas.

5. Ninguna es correcta.

4. Complete la siguiente frase: “La Ingeniería de Software es una disciplina que buscar crear software a bajo costo y …………………………….” Justifique

5. ¿Qué es un modelo de procesos de desarrollos de software?

6. ¿Un proyecto de desarrollo de software se puede desarrollar con cualquier modelo de proceso de software?

7. Metodología Spotify es aplicable a cualquier tipo de organización

8. Que es un Guild/Gremio. Cuáles son sus beneficios.

**TEMA 1**

**Sobre la siguiente frase: “En el contexto de la ingeniería de software, un proceso es una prescripción rígida que garantiza elaborar software de calidad”. Diga si es Verdadero o Falso y Justifique:**

Falso. Un proceso en la ingeniería de software no es necesariamente una prescripción rígida. Los procesos pueden ser flexibles y adaptables a diferentes contextos y proyectos. La calidad del software no está garantizada únicamente por seguir un proceso rígido, sino también depende de otros factores como la competencia del equipo, la calidad de los requisitos, y las herramientas utilizadas.

Explique la diferencia entre actividades estructurales y actividades sombrilla. De un ejemplo de cada una:

Actividades estructurales son aquellas que forman parte del núcleo del proceso de desarrollo de software y se realizan de manera secuencial o iterativa. Ejemplo: Codificación.

Actividades sombrilla son actividades que se realizan de manera transversal durante todo el proceso de desarrollo y no pertenecen a una fase específica. Ejemplo: Gestión de configuración.

**¿Cuáles son las actividades de un proceso de desarrollo RUP según Roger Pressman?**

Modelado del negocio, Ingeniería de Requisitos, Análisis y diseño, codificación, Integración, Pruebas, Despliegue, Actividades Sombrilla.

Las otras opciones omiten fases importantes o agrupan incorrectamente las actividades.

**Complete la siguiente frase: “La Ingeniería de Software es una disciplina que busca crear software a bajo costo y …………………………….” Justifique:**

"La Ingeniería de Software es una disciplina que busca crear software a bajo costo y de alta calidad."

Justificación: La Ingeniería de Software se enfoca en la eficiencia y efectividad en la producción de software. Esto incluye minimizar costos y maximizar la calidad para satisfacer las necesidades del usuario final y mantener la competitividad en el mercado.

**¿Qué es un modelo de procesos de desarrollo de software?**

Un modelo de procesos de desarrollo de software es una estructura o guía que describe las fases y actividades necesarias para desarrollar software. Proporciona un marco para planificar, organizar y controlar el proceso de desarrollo.

**¿Un proyecto de desarrollo de software se puede desarrollar con cualquier modelo de proceso de software?**

No, no todos los proyectos de software se pueden desarrollar con cualquier modelo de proceso. La elección del modelo depende de varios factores como el tipo de proyecto, el tamaño del equipo, los requisitos, y el entorno del cliente. Modelos diferentes son más adecuados para diferentes tipos de proyectos. Por ejemplo, el modelo en cascada es adecuado para proyectos con requisitos bien definidos, mientras que Agile es mejor para proyectos con requisitos que cambian con frecuencia.

**¿Metodología Spotify es aplicable a cualquier tipo de organización?**

No, la metodología Spotify no es aplicable a cualquier tipo de organización. Esta metodología está diseñada para organizaciones que tienen una cultura y estructura que soportan equipos autónomos y un alto nivel de flexibilidad. Organizaciones con estructuras más tradicionales y jerárquicas pueden encontrar difícil implementar este enfoque sin realizar cambios significativos en su cultura y estructura.

**¿Qué es un Guild/Gremio? ¿Cuáles son sus beneficios?**

Un Guild o Gremio es un grupo de personas que comparten un interés o área de conocimiento común dentro de una organización, y que se reúnen regularmente para intercambiar ideas, prácticas y mejorar sus habilidades.

Beneficios:

Facilita el intercambio de conocimientos y experiencias entre los miembros.

Promueve la mejora continua y la innovación.

Fomenta la colaboración y el aprendizaje entre equipos que de otra manera podrían estar aislados.

**TEMA 2**

Ayuda a estandarizar prácticas y herramientas dentro de la organización.

1. Sobre la siguiente frase de Roger Pressman: “El software es elemento de un sistema lógico y no de uno físico”. Diga si es Verdadero o Falso y Justifique

2. ¿Porque surge la disciplina Ingeniería de Software como una nueva Ingeniería durante la crisis del software?

3. Una estructura de proceso general para la ingeniería de software consta de las siguientes actividades:

a. Comunicación. b. Planeación. C. Modelado, D. Construcción. E. Despliegue. F. Todas son Correctas

4. ¿La gestión de un Proyecto garantiza el éxito de este?

5. ¿Cuáles son las actividades de un proceso de desarrollo RUP según Roger Pressman?

a. Requerimientos, Análisis y diseño, codificación, Integración, Pruebas, Despliegue, Mantenimiento

b. Análisis y diseño, codificación, Integración, Pruebas, Despliegue.

c. Modelado del negocio, Ingeniería de Requisitos, Análisis y diseño, codificación, Integración, Pruebas, Despliegue, Actividades Sombrilla.

d. Todas son correctas.

e. Ninguna es correcta.

6. Complete la siguiente frase: “La Ingeniería de Software es una disciplina que buscar crear software a de Calidad y …………………………….” Justifique

7. Mencione 4 diferencias entre las Metodologías Scrum y Spotify.

8. ¿El «Chapter Lead» en la metodología Scrum que tareas debe realizar?

9. Que implica la actividad de la Ingeniería de Software Análisis

10. Si tuviera que desarrollar una aplicación que analice cada curso ofrecido en la universidad y reporte las calificaciones promedio obtenidas en el curso (por un determinado periodo), que modelo de proceso de desarrollo utilizaría. Justifique la respuesta.

**TEMA 2**

Sobre la siguiente frase de Roger Pressman: “El software es elemento de un sistema lógico y no de uno físico”. Diga si es Verdadero o Falso y Justifique:

Verdadero. El software es un conjunto de instrucciones lógicas que se ejecutan en el hardware (elemento físico). Es intangible y consiste en código y datos que dirigen el comportamiento del hardware para realizar tareas específicas.

**¿Por qué surge la disciplina Ingeniería de Software como una nueva Ingeniería durante la crisis del software?**

La disciplina de Ingeniería de Software surgió durante la crisis del software en la década de 1960 debido a problemas significativos con el desarrollo de software: costos altos, plazos incumplidos, baja calidad, y dificultades para gestionar la complejidad del software. Esta nueva disciplina se enfocó en aplicar principios de ingeniería para abordar estos problemas, mejorar la calidad, la previsibilidad y la productividad en el desarrollo de software.

Una estructura de proceso general para la ingeniería de software consta de las siguientes actividades:

**F. Todas son Correctas.**

Comunicación

Planeación

Modelado

Construcción

Despliegue

**¿La gestión de un Proyecto garantiza el éxito de este?**

No, la gestión de un proyecto no garantiza el éxito, pero aumenta significativamente las probabilidades de éxito. La gestión efectiva incluye la planificación, organización, control y monitoreo del proyecto, lo cual ayuda a mitigar riesgos, optimizar recursos y cumplir con los objetivos del proyecto.

**¿Cuáles son las actividades de un proceso de desarrollo RUP según Roger Pressman?**

La respuesta correcta es:

c. Modelado del negocio, Ingeniería de Requisitos, Análisis y diseño, codificación, Integración, Pruebas, Despliegue, Actividades Sombrilla.

Las otras opciones no incluyen todas las fases del RUP (Rational Unified Process) según lo descrito por Roger Pressman.

**Complete la siguiente frase: “La Ingeniería de Software es una disciplina que buscar crear software a de Calidad y …………………………….” Justifique:**

"La Ingeniería de Software es una disciplina que busca crear software de calidad y a bajo costo."

Justificación: Además de la calidad, la Ingeniería de Software también se enfoca en la eficiencia económica. Esto implica minimizar los costos de desarrollo y mantenimiento mientras se asegura que el software cumpla con los requisitos y estándares de calidad.

**Mencione 4 diferencias entre las Metodologías Scrum y Spotify:**

Organización de Equipos:

Scrum: Equipos pequeños y multifuncionales que trabajan en sprints.

Spotify: Equipos llamados "squads" que son autónomos y trabajan en misiones específicas.

Estructura de Escalamiento:

Scrum: Usa "Scrum of Scrums" para coordinar equipos múltiples.

Spotify: Usa "tribus" para agrupar varios squads que trabajan en áreas relacionadas.

Roles:

Scrum: Roles específicos como Product Owner, Scrum Master, y equipo de desarrollo.

Spotify: Roles adicionales como "Chapter Leads" y "Guild Coordinators" que cruzan squads.

Enfoque en la Cultura:

Scrum: Enfocado en los valores y principios de Scrum.

Spotify: Enfocado en la cultura y autonomía de los equipos para fomentar la innovación.

**¿El «Chapter Lead» en la metodología Scrum que tareas debe realizar?**

En realidad, el rol de "Chapter Lead" no existe en la metodología Scrum; es un rol de la metodología Spotify. En Spotify, un "Chapter Lead" lidera un grupo de personas con habilidades similares que pertenecen a diferentes squads. Tareas incluyen:

Proveer mentoría y desarrollo profesional a los miembros del chapter.

Asegurar la consistencia en las prácticas y estándares técnicos.

Colaborar con otros Chapter Leads para la mejora continua de prácticas y herramientas.

**¿Qué implica la actividad de la Ingeniería de Software Análisis?**

La actividad de Análisis en la Ingeniería de Software implica comprender y definir los requisitos del usuario y del sistema, documentar las necesidades y restricciones del proyecto, y elaborar un modelo detallado que sirva como base para el diseño y desarrollo del software. Esto incluye el análisis de requisitos, modelado de datos y procesos, y la identificación de soluciones viables.

**Si tuviera que desarrollar una aplicación que analice cada curso ofrecido en la universidad y reporte las calificaciones promedio obtenidas en el curso (por un determinado periodo), ¿qué modelo de proceso de desarrollo utilizaría? Justifique la respuesta:**

Utilizaría el modelo Ágil.

Justificación: El modelo Ágil permite una entrega rápida y continua de pequeñas partes funcionales del software, lo cual es ideal para proyectos con requisitos cambiantes o cuando se necesita retroalimentación constante de los usuarios. Dado que la aplicación universitaria puede requerir ajustes frecuentes basados en las necesidades del usuario y la evolución de los datos y reportes, la flexibilidad y adaptabilidad del modelo Ágil serían beneficiosas. Además, permite iteraciones cortas y frecuentes, asegurando que el proyecto se mantenga alineado con las expectativas del cliente y puede adaptarse rápidamente a nuevos requerimientos.