1. DATOS INFORMATIVOS

Carrera: Ingenieria en Tecnologias De La Información

Asignatura: Metodología de Desarrollo en Software

Docente: Jenny Ruiz Robalino

Integrantes:

Nayely Simbaña, Sebastián Medina, Benjamín López

Fecha: 17/11/2025 **Nrc:** 29022

Actividad 1: Exploración Guiada (10 min)

Objetivo: Leer y comprender los fundamentos del RUP.

Instrucciones:

1. Lee el siguiente fragmento del Capítulo IX sobre RUP.

2. Extrae: - Las 4 fases principales del RUP.

- **Fase de Inicio (Inception):** Su objetivo principal es delimitar el alcance del proyecto, identificar los riesgos principales y establecer un caso de negocio viable. "Se enfoca en la captura de requisitos y la definición de la visión del producto" (Jacobson, Booch & Rumbaugh, 2000, Cap. IX).
- **Fase de Elaboración (Elaboration):** En esta fase se analizan los requisitos en detalle y se establece una arquitectura base sólida para el sistema. Se crea un plan de proyecto y se eliminan los riesgos técnicos más críticos. Su objetivo es desarrollar la comprensión del problema en el entorno del dominio, es decir, entender y abstraer del entorno el problema a resolver (Jacobson et al., 2000, Cap. IX), (Ruiz, J, 2025).
- **Fase de Construcción (Construction):** Es la fase de desarrollo principal, donde se construye el producto de forma iterativa hasta tener una versión operativa. La construcción implica desarrollar la mayor parte del software y preparar el producto para su transición. "El enfoque está en el desarrollo de componentes y otras características del sistema" (Jacobson et al., 2000, Cap. IX).
- **Fase de Transición (Transition):** La fase final se centra en llevar el producto al usuario final, lo que incluye actividades como capacitación y ajustes basados en la

retroalimentación. El producto es entregado a la comunidad de usuarios para su despliegue, se centra en entregar el sistema a los usuarios y asegurar su aceptación (Jacobson et al., 2000, Cap. IX).

2. Diferencias entre disciplinas de desarrollo y disciplinas de soporte.

El RUP estructura sus actividades en "disciplinas". Las disciplinas de desarrollo se centran en la producción directa del software, mientras que las de soporte describen cómo se mantiene el proceso bajo control.

1. Enfoque y Resultado:
    - Desarrollo: Se concentran en la construcción y definición técnica del producto. Por ejemplo, la disciplina de Modelado de Negocio define procesos, y la de Requisitos captura lo que el sistema debe hacer.
    - Soporte: Se centran en las actividades de gestión y control del proceso de desarrollo. Por ejemplo, Gestión de Configuración y Cambios controla las versiones del software, y Gestión de Proyecto planifica y supervisa el progreso (Jacobson et al., 2000, Cap. IX).
2. Naturaleza de las Actividades:
    - Desarrollo: Son actividades centrales y creativas que añaden valor directo al producto final, como el diseño, la implementación y las pruebas del software.
    - Soporte: Son actividades facilitadoras y de garantía de calidad que aseguran que el proceso de desarrollo sea ordenado, controlado y de alta calidad, pero no producen código o modelos directamente (Jacobson et al., 2000, Cap. IX).

Ventajas del enfoque iterativo e incremental

3. Ventajas del Enfoque Iterativo e Incremental

El RUP se fundamenta en un desarrollo iterativo e incremental, lo que reporta varias ventajas significativas frente a un enfoque lineal en cascada.

- **Gestión Proactiva de Riesgos:** "Los riesgos se atacan desde las primeras iteraciones, lo que permite identificarlos y mitigarlos mucho antes" (Jacobson et al., 2000, Cap. IX). Al abordar los aspectos más arriesgados al inicio (por ejemplo, en la fase de Elaboración), se reduce la probabilidad de fracaso del proyecto en etapas tardías.
- **Mayor Adaptabilidad a Cambios:** Este enfoque permite incorporar feedback de los stakeholders de manera continua. "Los cambios en los requisitos pueden ser acomodados más fácilmente, ya que cada iteración ofrece una oportunidad para ajustar la dirección del proyecto" (Jacobson et al., 2000, Cap. IX). Esto es crucial en

entornos donde los requisitos son inestables o no están completamente definidos al inicio.

- **Mejora Continua y Aprendizaje:** El equipo aprende y mejora continuamente a lo largo del proyecto. Cada iteración sirve como un ciclo de retroalimentación que permite refinar la arquitectura, los planes y las competencias del equipo. "El proyecto puede beneficiarse de las lecciones aprendidas en iteraciones anteriores" (Jacobson et al., 2000, Cap. IX).