

**INSTITUTO SUPERIOR DE FORMACION TECNICA N°130**

**CARRERA**

ANALISTA DE SISTEMAS

-

**MATERIA**

SISTEMAS Y ORGANIZACIONES

-

**PRIMER AÑO**

**TRABAJO PRACTICO N° 1**

-

**DOCENTE**

PABLO LETIER

-

**ALUMNO**

EMILIANO SEQUEIRA

LAUTARO CASTILLO

EZEQUIEL KOLMAN

SAUL HORACIO SOSA PONCETTA

**AÑO**

2024

**TRABAJO PRÁCTICO Nº1**

**Temas:**

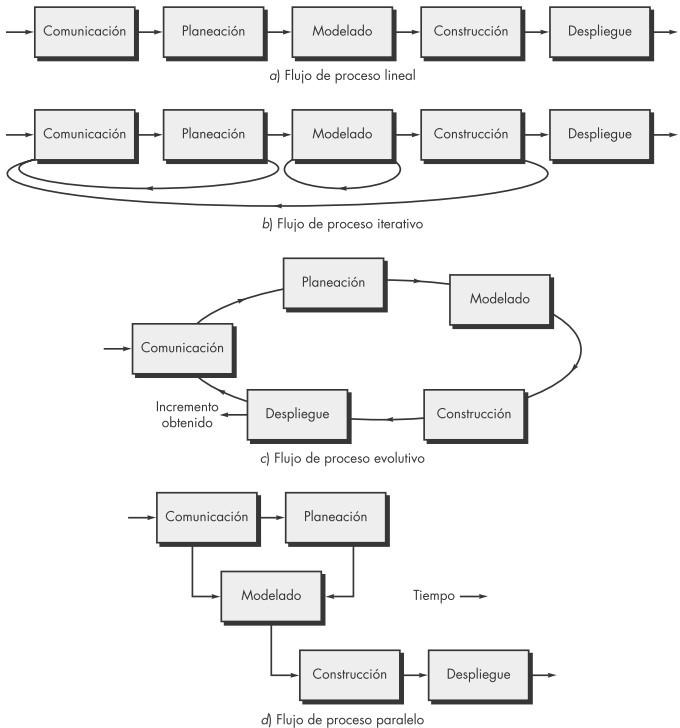
1. Modelos prescriptivos de proceso de desarrollo de software.
2. La ingeniería de software ágil.

**Descripción:**

1. Modelos prescriptivos de proceso de desarrollo de software.

Los modelos prescriptivos de proceso definen un conjunto distinto de actividades, acciones, tareas, fundamentos y productos de trabajo que se requieren para desarrollar software de alta calidad. Los ingenieros de software han elegido de manera tradicional un marco de trabajo genérico para el proceso, el cual incluye las siguientes actividades estructurales: comunicación, planeación, modelado, construcción y desarrollo.

El *flujo del proceso* describe la manera en que están organizadas las actividades estructurales y las acciones y tareas que ocurren dentro de cada una con respecto de la secuencia y el tiempo. Según la metodología a utilizar, encontramos diferentes tipos de flujos de proceso:



Modelos de proceso de software:

* + Modelos de Cascada (Waterfall Model)
  + Modelos de Proceso Incremental (Incremental Models)
  + Modelos de Proceso Evolutivo (Evolutionary Process Models)
  + Modelos de Proceso Especializado
  + Proceso Unificado (Unified Process)

1. La ingeniería de software ágil.

La ingeniería de software ágil combina una filosofía y un conjunto de directrices de desarrollo. La filosofía busca la satisfacción del cliente y la entrega temprana de software incremental, equipos de proyecto pequeños y altamente motivados, un mínimo de productos de trabajo de la ingeniería de software y simplicidad general en el desarrollo. Las directrices de desarrollo resaltan la entrega sobre el análisis y el diseño, y la comunicación activa entre los desarrolladores y los clientes.

Modelos de procesos ágiles:

* + Programación Extrema (Extreme Programming) (XP)
  + Desarrollo Adaptativo de Software (Adaptive Software Development) (ASD)
  + Método de Desarrollo de Sistemas Dinámicos (Dynamic Systems Development Method) (DSDM)
  + Scrum
  + Cristal (Crystal)
  + Desarrollo Impulsado por las Características (Feature Driven Development) (FDD)
  + Modelado Ágil (Agile Modeling) (AM)
  + Desarrollo de Software Lean (Lean Software Development)

**Actividades a realizar (grupos de 5 integrantes como máximo):**

1. Lectura de referencia: capítulos 2 y 3 del libro “Ingeniería del Software” Pressman Roger, ed. Mc Graw Hill, séptima edición. Nota: El ejemplar en formato digital se encuentra disponible en la sección Recursos del sitio de aulas virtuales.
2. Buscar fuentes de información adicionales sobre el tema asignado al grupo: bibliografía, textos en Internet, documentación en formato digital, etc.
3. Entregar un informe en formato digital (Word) sobre el modelo de desarrollo asignado (máximo 5 carillas fuente arial 10). Este informe debe contener, **obligatoriamente**:
   1. descripción de la metodología,
   2. filosofía adoptada o principales principios,
   3. Presentación gráfica del proceso con su explicación correspondiente.
4. Realizar una presentación en clase de una duración **entre 10 y 12 minutos**, basada en la investigación realizada. Para utilizar como apoyo de la exposición, puede emplearse una presentación en Power Point, Prezi o cualquier otra herramienta similar. Se dispondrá del proyector y una notebook del Instituto, que los grupos podrán utilizar para su exposición.

**Fecha de entrega:**

Tanto el informe en Word como la presentación (Power Point) deben ser entregados en formato digital a través de las aulas virtuales, con fecha límite el día sábado 22 de junio de 2024. Cada grupo tiene asignado un horario en el cual realizará su exposición en clase, el día lunes 24 de junio. Por cuestiones de tiempo, si quedaran grupos sin poder exponer, se continúa con las presentaciones grupales el día lunes 8 de julio.

**Evaluación:**

En la evaluación de los trabajos se contemplará:

* Fuentes de información y material recolectado.
* Organización y calidad de presentación del trabajo.
* Contenidos del informe presentado.
* Análisis y conclusiones sobre la información obtenida.
* Calidad y claridad de la exposición en clase

Cumplimiento del tiempo máximo asignado a la exposición en clase Consideraciones generales para organizar la exposición en clase:

* + Plantear título de la metodología
  + Ventajas y desventajas
  + Tipos de proyecto a los cuales se ajusta mejor
  + Diapositiva: mostrar títulos o frases importantes, y ampliar texto oralmente
  + Gráficos explicativos, conceptuales, para mejorar presentación visual
  + Utilizar letra visible (grande) pensando en la distancia del auditorio
  + Practicar la exposición y ajustar los defectos que se visualicen
  + **No leer**, se puede utilizar un texto de apoyo pero no todo leído y recitado!

Nota: Existen en Internet muchísimos cursos y ayudas sobre cómo realizar correctamente exposiciones en Power Point. Les sugiero que consulten estos sitios, y apliquen estas recomendaciones en sus trabajos.

Recomendación: si bien estará disponible la notebook del Instituto, les recomiendo que los alumnos el día de la exposición lleven sus propios equipos con los cuales hayan practicado la presentación, para evitar inconvenientes e incompatibilidades a la hora de realizar la misma.

**Proceso de ingeniería de software**

**¿Qué es?** —el mapa de carreteras que lo ayuda a obtener a tiempo un resultado de alta calidad—. El mapa que se sigue se llama “proceso del software”

**¿Quién lo hace?** Los ingenieros de software y sus gerentes adaptan el proceso a sus necesidades y luego lo siguen.

**¿Por qué es importante?** Porque da estabilidad, control y organización a una actividad que puede volverse caótica si se descontrola. Sin embargo, un enfoque moderno de ingeniería de software debe ser “ágil”.

**¿Cuáles son los pasos?** En un nivel detallado, el proceso que se adopte depende del software que se esté elaborando. Un proceso puede ser apropiado para crear software destinado a un sistema de control electrónico de un aeroplano

**¿Cuál es el producto final?** Los productos del trabajo son los programas, documentos y datos que se producen como consecuencia de las actividades y tareas definidas por el proceso.

**¿Cómo me aseguro de que lo hice bien?** La calidad, oportunidad y viabilidad a largo plazo del producto que se elabora son los mejores indicadores de la eficacia del proceso que se utiliza.

Una estructura general para la ingeniería de software define cinco actividades estructurales:

* comunicación
* planeación
* modelado
* construcción
* despliegue

Un **flujo de proceso lineal** ejecuta cada una de las cinco actividades estructurales en secuencia, comenzando por la comunicación y terminando con el despliegue

Un **flujo de proceso iterativo** repite una o más de las actividades antes de pasar a la siguiente

Un **flujo de proceso evolutivo** realiza las actividades en forma “circular”. A través de las cinco actividades, cada circuito lleva a una versión más completa del software

Un **flujo de proceso paralelo** ejecuta una o más actividades en paralelo con otras (por ejemplo, el modelado de un aspecto del software tal vez se ejecute en paralelo con la construcción de otro aspecto del software).

**MODELOS DE PROCESO PRESCRIPTIVO**

Los modelos de proceso prescriptivo fueron propuestos originalmente para poner orden en el caos del desarrollo de software

Cada modelo del proceso también prescribe un flujo del proceso (también llamado flujo de trabajo), es decir, la manera en la que los elementos del proceso se relacionan entre sí. Todos los modelos del proceso del software pueden incluir las actividades estructurales generales descritas en el capítulo, pero cada una pone distinto énfasis en ellas y define en forma diferente el flujo de proceso que invoca cada actividad estructural (así como acciones y tareas de ingeniería de software).

### Modelo de Cascada (Waterfall Model)

#### Descripción

El Modelo de Cascada creado en 1970 por el programador y teórico informático Winston W.Royce, también conocido como Waterfall Model, es uno de los enfoques más tradicionales y ampliamente reconocidos en el desarrollo de software. Se caracteriza por su naturaleza secuencial y lineal, donde cada fase del proceso de desarrollo debe completarse por completo antes de pasar a la siguiente. Este modelo toma su nombre de la analogía con una cascada de agua, en la que el flujo desciende de un nivel a otro de manera descendente y unidireccional.

En la práctica, esto significa que el equipo de desarrollo no avanza a la siguiente etapa sin haber finalizado completamente la actual. Por ejemplo, los requisitos deben estar totalmente definidos y documentados antes de proceder al diseño del sistema, el diseño debe estar completamente finalizado antes de comenzar la implementación, y así sucesivamente. Este enfoque pretende asegurar que cada fase se complete a fondo y que los entregables de cada etapa sean aprobados antes de avanzar.

#### Filosofía

El Modelo de Cascada adopta una filosofía de desarrollo ordenado y disciplinado. Está basado en la premisa de que un enfoque sistemático y riguroso minimiza los riesgos y asegura que el producto final cumpla con las especificaciones y requisitos iniciales. Esta metodología pone un fuerte énfasis en la documentación detallada y en el cumplimiento de cada etapa del desarrollo, promoviendo una estructura clara y predecible.

La filosofía subyacente del Modelo de Cascada se basa en la idea de que los proyectos de software pueden planificarse y ejecutarse de manera similar a otros proyectos de ingeniería, con fases bien definidas y resultados esperados claramente especificados. Se asume que los requisitos son bien entendidos y estables desde el principio del proyecto, lo que permite una planificación y ejecución rigurosas.

#### 

#### Etapas

El Modelo de Cascada se divide en cinco etapas principales:

**Comunicación.** Antes de que comience cualquier trabajo técnico, tiene importancia crítica comunicarse y colaborar con el cliente (y con otros participantes). Se busca entender los objetivos de los participantes respecto del proyecto, y reunir los requerimientos que ayuden a definir las características y funciones del software.

* + **Objetivo**: Capturar y documentar todas las necesidades y expectativas del cliente. Capturar los requerimientos
  + **Actividades**: Entrevistas con usuarios, análisis de requisitos, creación de especificaciones de requisitos de software (SRS).
  + **Productos**: Documentos de requisitos, especificaciones funcionales y no funcionales.

**Planeación.** la actividad de planeación crea un “mapa” que guía al equipo. El mapa —llamado *plan del proyecto de software*— define el trabajo de ingeniería de software al describir las tareas técnicas por realizar, los riesgos probables, los recursos que se requieren, los productos del trabajo que se obtendrán y una programación de las actividades.

**Modelado.** Crea un “bosquejo” del objeto por hacer a fin de entender el panorama general —cómo se verá arquitectónicamente, cómo ajustan entre sí las partes constituyentes y muchas características más—. Si se requiere, refina el bosquejo con más y más detalles en un esfuerzo por comprender mejor el problema y cómo resolverlo. Un ingeniero de software hace lo mismo al crear modelos a fin de entender mejor los requerimientos del software y el diseño que los satisfará.

* + **Objetivo**: Convertir los requisitos en una arquitectura de sistema y un diseño detallado.
  + **Actividades**: Diseño de alto nivel (arquitectura del sistema), diseño de bajo nivel (detalles técnicos y componentes).
  + **Productos**: Diagramas de arquitectura, modelos de datos, especificaciones de diseño detallado.

**Construcción.** Esta actividad combina la generación de código (ya sea manual o automatizada) y las pruebas que se requieren para descubrir errores en éste.

* + **Objetivo**: Traducir el diseño en código ejecutable.
  + Validar y verificar que el software cumple con los requisitos y funciona correctamente.
  + **Actividades**: Codificación de módulos, integración de componentes, desarrollo de unidades de software.
  + Pruebas unitarias, pruebas de integración, pruebas del sistema, pruebas de aceptación
  + **Productos**: Código fuente, programas compilados.Planes de prueba, casos de prueba, resultados de pruebas, informes de defectos.

**Despliegue.** El software (como entidad completa o como un incremento parcialmente terminado) se entrega al consumidor que lo evalúa y que le da retroalimentación, misma que se basa en dicha evaluación.

* + **Objetivo**: Corregir errores, mejorar el rendimiento y adaptar el software a cambios en el entorno operativo o requisitos adicionales.
  + **Actividades**: Corrección de errores, actualizaciones, mejoras, gestión de versiones.
  + **Productos**: Parches de software, versiones actualizadas, documentación de cambios.

#### Ventajas

El Modelo de Cascada presenta varias ventajas que lo hacen atractivo, especialmente para ciertos tipos de proyectos:

1. **Simplicidad y Claridad**: Es fácil de entender y aplicar debido a su enfoque lineal y secuencial.
2. **Estructura Ordenada:** Proporciona una estructura clara y bien definida para cada etapa del desarrollo.
3. **Documentación Detallada:** Promueve la creación de una documentación exhaustiva en cada fase, lo que facilita la revisión y el mantenimiento del proyecto.
4. **Control y Gestión:** Permite un control y gestión del proyecto más estrictos, con entregables claramente definidos en cada etapa.
5. **Adecuado para Proyectos Pequeños:** Es particularmente útil en proyectos pequeños con requisitos bien entendidos y poco susceptibles a cambios.

#### Desventajas

A pesar de sus ventajas, el Modelo de Cascada tiene varias desventajas significativas:

1. **Rigidez y Falta de Flexibilidad**: Su naturaleza secuencial lo hace poco flexible a cambios en los requisitos una vez que una fase ha sido completada. Es necesaria una gestión estricta y un seguimiento periodico,para que el proyecto cumpla los pasos.
2. **Dificultad para retroceder**: Es complicado y costoso volver a fases anteriores para realizar modificaciones o correcciones, lo que puede generar problemas si se descubren errores tardíos.
3. **Suposiciones de Requisitos Estáticos:** Asume que todos los requisitos pueden ser completamente definidos al inicio del proyecto, lo cual no siempre es realista.
4. **Riesgo de Problemas No Detectados:** Los errores y problemas pueden no ser detectados hasta etapas tardías del desarrollo, incrementando el riesgo de costosos retrabajos.
5. **Menos Adaptabilidad:** No se adapta bien a proyectos donde los requisitos son inciertos o propensos a cambios durante el desarrollo.

**Conclusión**:

El Modelo de Cascada es un enfoque de desarrollo de software secuencial y bien estructurado que puede ser efectivo en contextos específicos. Sin embargo, su falta de flexibilidad y adaptabilidad a cambios en los requisitos lo hace menos adecuado para proyectos dinámicos y complejos donde la iteración y la retroalimentación continua son críticas.

