Nombre: & Gabriel Méndez
C.I.: & V-30.464.232
Materia: & Análisis de Sistemas
Sección: & DCM0402
Universidad: & UniversidadAlejandroDeHumboldt
Carrera: & IngenieriaInformatica
Evaluación: & Evaluación N°1 – Corte 1
Fecha: & 11/02/2025

# Materia: Análisis de Sistemas

# Introducción

El desarrollo de software ha evolucionado considerablemente en las últimas décadas, dando lugar a diversas metodologías, desde los enfoques tradicionales como el modelo en cascada hasta las metodologías ágiles como Scrum y XP. Cada uno presenta fortalezas y debilidades inherentes. En este documento, exploraremos un modelo de desarrollo híbrido que busca combinar lo mejor de ambos mundos: la planificación estructurada de las metodologías tradicionales con la flexibilidad y adaptabilidad de las metodologías ágiles. El objetivo es presentar un esquema que permita gestionar proyectos de software complejos, manteniendo un control adecuado del proceso y a la vez permitiendo responder eficientemente a los cambios en los requisitos.

# Desarrollo del Modelo Híbrido

El modelo propuesto se denomina Çascada Ágilz se fundamenta en una estructura de alto nivel tipo cascada, dividiendo el proyecto en fases bien definidas (Requisitos, Diseño, Implementación, Pruebas, Despliegue, Mantenimiento). Sin embargo, la clave reside en la forma en que se gestiona cada una de estas fases. Dentro de cada fase, se aplicarán principios y prácticas ágiles, principalmente de Scrum y XP, para llevar a cabo las tareas específicas.

## Fase de Requisitos (Tradicional + Ágil)

Esta fase se inicia con la recolección inicial de requisitos de manera tradicional, utilizando entrevistas, documentación existente y análisis de la necesidad del negocio. Se crea un documento de requisitos inicial de alto nivel. Luego, se utiliza Scrum para refinar estos requisitos de manera iterativa.

* **Scrum:** Se crean sprints cortos (de 1 a 2 semanas) dedicados a la especificación y priorización de requisitos. El Product Owner (representante del cliente) trabaja estrechamente con el equipo de desarrollo para refinar las historias de usuario, definir los criterios de aceptación y priorizar el backlog del producto. Se utilizan técnicas de priorización como MoSCoW (Must have, Should have, Could have, Won't have) para determinar

qué requisitos son esenciales para el proyecto. * **Resultado:** Un backlog del producto detallado y priorizado, con historias de usuario claras y criterios de aceptación definidos.

## Fase de Diseño (Tradicional + XP)

Se utiliza el diseño inicial de la fase de requisitos como base para crear una arquitectura general del sistema. Dentro de esta arquitectura, se emplean prácticas de XP para el diseño detallado y la implementación de cada funcionalidad.

* **XP (Programación Extrema):** * **Diseño simple:** Se busca el diseño más sencillo que cumpla con los requisitos. Se evita la sobre-ingeniería. * **Programación en parejas:** Dos programadores trabajan juntos en una misma estación de trabajo, intercambiando ideas y revisando el código en tiempo real. * **Refactorización:** Se mejora continuamente el código, eliminando la duplicación y mejorando la legibilidad y la estructura. * **Arquitectura:** Se define la arquitectura inicial, los componentes principales y las interfaces. Se pueden utilizar diagramas UML para representar la arquitectura.

## Fase de Implementación (Ágil)

Esta fase se basa completamente en Scrum.

* **Scrum:** El equipo de desarrollo trabaja en sprints para implementar las historias de usuario del backlog. En cada sprint, se planifica el trabajo, se ejecutan las tareas, se realiza una revisión del sprint (demo) y una retrospectiva para mejorar el proceso. * **Integración continua:** Se integra el código de forma frecuente (varias veces al día) para detectar errores tempranamente. * **Pruebas unitarias:** Se escriben pruebas unitarias para verificar que cada componente funcione correctamente.

## Fase de Pruebas (Tradicional + Ágil)

Se combinan pruebas tradicionales (pruebas de sistema, pruebas de aceptación) con pruebas ágiles (pruebas automatizadas, pruebas exploratorias).

* **Pruebas automatizadas:** Se crean scripts de prueba automatizados para verificar la funcionalidad del sistema de forma repetitiva y eficiente. * **Pruebas exploratorias:** Los testers exploran el sistema buscando errores inesperados. * **Pruebas de aceptación:** El Product Owner valida que el sistema cumple con los criterios de aceptación de las historias de usuario.

## Fase de Despliegue (Tradicional + Ágil)

El despliegue se realiza de forma controlada, utilizando un plan de despliegue detallado. Sin embargo, se puede utilizar la entrega continua (parte de DevOps) para automatizar el proceso de despliegue y hacerlo más frecuente y eficiente.

## Fase de Mantenimiento (Ágil)

El mantenimiento se gestiona utilizando Scrum, priorizando los errores y las nuevas funcionalidades en el backlog del producto.

## Ventajas del Modelo Cascada Ágil

* **Control y visibilidad:** La estructura en cascada proporciona un marco de control claro y una mayor visibilidad del progreso del proyecto. * **Adaptabilidad:** El uso de metodologías ágiles dentro de cada fase permite responder a los cambios en los requisitos de forma flexible. * **Reducción de riesgos:** La integración continua y las pruebas frecuentes ayudan a detectar y corregir errores tempranamente, reduciendo los riesgos del proyecto. * **Mejora continua:** Las retrospectivas de Scrum permiten al equipo aprender de sus errores y mejorar continuamente el proceso de desarrollo. * **Adecuado para proyectos complejos:** Este modelo es particularmente adecuado para proyectos complejos que requieren una planificación detallada y un control estricto.

## Desafíos del Modelo Cascada Ágil

* **Complejidad:** La combinación de metodologías tradicionales y ágiles puede aumentar la complejidad del proyecto. * **Resistencia al cambio:** Los miembros del equipo que están acostumbrados a trabajar con metodologías tradicionales pueden resistirse a adoptar las prácticas ágiles. * **Comunicación:** Es fundamental una comunicación clara y constante entre todos los miembros del equipo para evitar malentendidos. * **Gestión de la transición:** La transición a un modelo híbrido puede ser un desafío, y requiere una planificación cuidadosa y un compromiso por parte de todos los involucrados. * **Adaptación de las herramientas:** Es posible que se necesiten herramientas de gestión de proyectos que soporten tanto las metodologías tradicionales como las ágiles.

## Ejemplo Práctico

Consideremos un proyecto de desarrollo de una aplicación web para una empresa de comercio electrónico.

1. **Requisitos:** Se recopilan los requisitos iniciales con el cliente. El Product Owner crea un backlog del producto con historias de usuario como: Çomo cliente, quiero poder buscar productos por nombre, categoría y precio". 2. **Diseño:** Se define la arquitectura de la aplicación, incluyendo la base de datos, la interfaz de usuario y los servicios web. Se utiliza programación en parejas para diseñar las clases y los métodos que implementan la funcionalidad de búsqueda. 3. **Implementación:** El equipo de desarrollo trabaja en sprints de dos semanas para implementar las historias de usuario del backlog. En cada sprint, se implementa una o varias historias de usuario, se realizan pruebas unitarias y se integra el código. 4. **Pruebas:** Se realizan pruebas automatizadas para verificar que la funcionalidad de búsqueda funciona correctamente. También se realizan pruebas exploratorias para buscar errores inesperados. 5. **Despliegue:** Se despliega la aplicación en un entorno de prueba para que el cliente pueda revisarla. Luego se despliega en el entorno de producción. 6. **Mantenimiento:** Se utilizan Scrum para gestionar los errores y las nuevas funcionalidades que se identifican después del despliegue.

# Conclusión

El modelo Çascada Ágilrepresenta un intento de aprovechar las fortalezas de las metodologías tradicionales y ágiles, creando un marco de desarrollo que permite un control adecuado del proyecto y una respuesta flexible a los cambios. Si bien presenta desafíos, los beneficios potenciales en términos de control, adaptabilidad y calidad del producto lo convierten en una opción viable para proyectos de software complejos. La clave del éxito radica en una planificación cuidadosa, una comunicación efectiva y un compromiso por parte de todos los involucrados en el proyecto.

# Referencias Bibliográficas

* Schwaber, K., & Sutherland, J. (2020). *The Scrum Guide*. Scrum.org. * Beck, K. (2000). *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional. * Royce, W. W. (1970). Managing the development of large software systems. *Proceedings of IEEE WESCON*, 1-9. * Sommerville, I. (2016). *Software Engineering* (10th ed.). Pearson Education. * Pressman, R. S., & Maxim, B. R. (2015). *Software Engineering: A Practitioner's Approach* (8th ed.). McGraw-Hill Education.