INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

INGENIERIA DE SOFTWARE

Méndez Segundo Laura

TAREA 3

CUADRO COMPARATIVO DE MODELOS DE PROCESO Y METODOLOGÍAS

Arellano Aguillón Shu Nashy Nizarely
Escudero Robles Rafael Agustín
Ramírez Rodríguez Carlos Eduardo

3CM13

22 / MARZO / 2021

NOMBRE	AUTOR	CARACTERÍSTICAS PRINCIPALES (5)	ETAPAS	VENTAJAS	DESVENTAJAS	TIPO DE DESARROLLOS EN LOS QUE SE APLICA
Modelo por prototipos	HASSAN GOMAA	• Se construyen prototipos los cuales se prueban y se modifican si es necesario • Se centran en la representación de aquellos aspectos que serán visibles para el cliente o usuario final • El prototipo debe ser construido en poco tiempo. • Se lleva a acabo en un equipo de desarrollo reducido • Se va modificando y desarrollando el sistema sobre la marcha	1. Investigación preliminar 2. Análisis y especificación 3. Diseño y construcción 4. Evaluación 5. Modificación 6. Diseño técnico 7. Programación y prueba 8. Operación y mantención	• No modifica el flujo de ciclo de vida • Reduce costos y aumenta la probabilidad de éxito. • Permite la modificación del sistema en etapas tempranas. • El desarrollador puede identificar lo que el cliente busca. • Permite que el cliente se dé cuenta de cómo está avanzando el proyecto. • Permite obtener retroalimentación de los usuario o clientes.	• Conlleva una administración difícil del desarrollo, ya que durante el ciclo de desarrollo se puede perder de vista cuál era el propósito inicial. • Se puede llegar a adoptar un prototipo como sistema final, incluyendo que el prototipo se puede encontrar incompleto o inadecuado. • El desarrollo y el cliente tiene poca comunicación al inicio del proceso. • Surgen cambios imprevistos que retrasan el progreso del prototipo.	Cuando un cliente define un conjunto de objetivos generales para el software, pero sin delimitar los requisitos de entrada, procesamiento y salida.
Modelo Incremental	HARLAN MILLS	• Las tareas están divididas en iteraciones • Cada iteración está relacionada con la iteración anterior suponiendo un avance a su iteración previa. • Es una combinación de forma secuencial e iterativo a través de prototipos funcionales • Tiene como objetivo un crecimiento progresivo de la funcionalidad • Se establecen entregas parciales mediante un calendario de plazos	1. Requerimientos 2. Definición de tareas y las iteraciones 3. Diseño de incrementos 4. Desarrollo de incrementos 5. Validación de incrementos 6. Integración de incrementos 7. Entrega del producto	• Permite una fácil administración de las tareas en cada iteración • La inversión se materializa a corto plazo • Es un modelo propicio a cambios o modificaciones • Se adapta a las necesidades que surjan	• No es recomendable para casos de sistemas de tiempo real, de alto nivel de seguridad o alto índice de riesgo. • Requiere mucha planeación tanto administrativa como técnica • Requiere metas claras para conocer el estado del proyecto	Cuando no se cuenta con una dotación de personal suficiente.
Modelo en Espiral	BARRY BOEHM	• Es una combinación entre el modelo lineal y el iterativo. • Consiste en seguir ciclos crecientes.	1. Determinar o fijar los objetivos 2. Análisis de Riesgo. 3. Ingeniería	• Integración de promotores y de usuarios en el proceso de Desarrollo. • Valoración periódica de los Riesgos	• Conlleva mucha documentación: es tedioso. • Existe el riesgo de que se formen Bucles.	En proyectos donde el coste de un fallo es un gran riesgo, de ahí que su principal aportación

		• Es considerado como un modelo evolutivo. • Es genérico (Puede combinarse con otros Modelos de desarrollo). • Se caracteriza por la minimización de riesgos.	4. Evaluación del cliente.	• Feedback por la estructura cíclica. • Apropiado para entornos novedosos.	• Hay errores e incongruencias al producto final por los prototipos. • No es apropiado para proyectos pequeños	sea considerar la gestión de riesgos.
Desarrollo Rápido de Aplicaciones	JAMES MARTIN	• Es un modelo que suelen englobar la usabilidad, utilidad y la rapidez de ejecución • Los desarrolladores deben ser analistas, diseñadores y programadores en uno • Las funciones secundarias son eliminadas como sea necesario para cumplir con el calendario • Prototipos iterativos y evolucionario • Equipos compuestos por alrededor de seis personas, incluyendo desarrolladores y usuarios de tiempo completo del sistema, así como aquellas personas involucradas con los requisitos.	1. Planificación de necesidades 2. Construcción 3. Transición	• Comentarios constantes de los usuarios • Integración temprana de sistemas • adaptabilidad	• Requiere sistemas modulares • Dificultad dentro de proyectos a gran escala • Exige mucha interactividad del usuario • Necesidad de desarrolladores experimentados	Proyectos en los que es posible una modularización clara.
Modelo en V	ALAN DAVIS	• Es similar al proceso en cascada. • Define los procedimientos de gestión de la calidad que lo acompañan. • Su estructura se asemeja a una V, de ahí su nombre. • El lado izquierdo de la V representa la descomposición de las necesidades y especificaciones del sistema. • El lado derecho de la V, representa la integración de las piezas y su verificación	Parte izquierda: 1. Definición de requerimientos. 2. Diseño funcional del sistema. 3. Diseño técnico del sistema. 4. Especificación de componentes. Una vez generado el código, el equipo sube por el lado derecho de la V: 5. Código 6. Pruebas unitarias	• La relación entre las etapas de desarrollo y los distintos tipos de pruebas facilitan la localización de fallos. • En un modelo sencillo y de fácil aprendizaje. • Es un modelo sencillo y de fácil aprendizaje. • Especifica bien los roles de los distintos tipos de pruebas a realizar. • Involucra al usuario en las pruebas.	• Es difícil que el cliente exponga explícitamente todos los requisitos. • El cliente debe tener paciencia debido a que obtendrá el producto al final del ciclo de vida. • Las pruebas pueden ser caras y, a veces, no lo suficientemente efectivas. • El producto final puede que no refleje todos los requisitos del usuario.	Proyectos donde se sepan los requerimientos desde el principio, y se tengan bien establecidos.

			7. Pruebas de componentes 8. Pruebas del sistema 9. Pruebas de aceptación			
XP Extreme Programming	KENT BECK	• Se basa en la comunicación, reutilización de código y realimentación. • Pruebas unitarias continuas • Programación en parejas. • Frecuente integración del equipo de programación con el cliente o usuario. • Refactorización del código	1. Planificación 2. Diseño 3. Codificación 4. Pruebas 5. Lanzamiento	• Da lugar a una programación sumamente organizada. • Cuenta con una tasa de errores muy pequeña • Permite ahorrar mucho tiempo y dinero. • El cliente tiene control de las prioridades • Fomenta la comunicación entre clientes y programadores.	• En caso de fallar, las comisiones son muy altas. • Requiere de un rígido ajuste a los principios de XP. • Puede no siempre ser más fácil que el desarrollo tradicional.	Es recomendable emplearla sólo en proyectos a corto plazo, y es mejor utilizada con la implementación de nuevas tecnologías.
Crystal	ALISTAIR COCKBURN	• Impulsado por el hombre, es decir, da importancia a los involucrados y sus necesidades. • Es adaptativo, los procesos y las herramientas se ajustan a los requerimientos y características del proyecto. • Es ultraligero, no hay documentación extensiva, sobre gestión y sin informes particulares. • Se compone de enfoques. • Seguridad personal.	1. Puesta en escena: Planificación 2. Revisiones 3. Monitoreo 4. Paralelismo y flujo 5. Estrategia de diversidad holística 6. Técnica de puesta a punto de la metodología 7. Puntos de vista de usuario	• Permite a los equipos trabajar en la manera que consideren más efectiva • Facilita la comunicación directa con el equipo. • El enfoque adaptativo permite a los equipos responder bien a cambios en los requerimientos.	• La falta de pre-planeación puede llevar a procesos fuera de alcance. • La falta de documentación puede llevar a confusión.	Puede ser usado en proyectos grandes y para gestionar proyectos ágiles.
Proceso Unificado Racional	IVAR JACOBSON Y JAMES RUMBAUGH	• Es iterativo e incremental. • Se basa en casos de uso. • Verifica de manera continua la calidad del software. • Administra los requisitos. • Es la implementación del desarrollo en espiral.	1. Inicio: Definir y acordar el alcance del proyecto. 2. Elaboración: Análisis y diseño. 3. Construcción: Construcción del producto. 4. Transición: Producto preparado para su entrega al usuario.	• Es el más general de los modelos actuales. • Forma disciplinada de asignar tareas y responsabilidades • Mantenimiento más sencillo y modificaciones locales • Reutilización	• Es un proceso bastante grande y complejo. • Es un modelo pesado. • Es bastante cara y con bastantes requerimientos en cuanto a roles.	Se puede utilizar desde el principio de un nuevo proyecto de software y se puede seguir utilizando en futuros proyectos relacionado al mismo. → Ciclo vital del proyecto

						→ Propósitos empresariales → Tamaño del esfuerzo de desarrollo de software
SCRUM	JEFF SUTHERLAND & KEN SCHWABER	• Está dividido en roles. • Se puede clasificar más como Framework • El valor principal es el coraje. • Es transparente. • Se requiere compromiso.	I. Divide & conquer II. Sprints III. Reuniones IV. Sprint backlog Sprint: 1. Sprint review. 2. Sprint retrospective. 3. Sprint Planning. 4. Daily Sprint.	• Scrum es muy fácil de aprender. • El cliente puede comenzar a usar el producto rápidamente. • Se agiliza el proceso. • Menor probabilidad de sorpresas o imprevistos.	• Es difícil de implementar. • La necesidad de tener equipos multidisciplinares puede ser un problema. • El equipo puede tender a realizar el camino más corto para conseguir el objetivo de un sprint	Está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos.
OMT	JAMES RUMBAUGH	• Se apoya en Modelos. • Trata de modelar el mundo real. • Utiliza diagramas de casos de uso. • Combinan estructuras de datos y comportamientos en una unidad simple (clases con atributos y métodos). • Se apoya de UML	1. Análisis 2. Diseño del sistema 3. Diseño de objetos 4. Implementación La metodología OMT emplea tres clases de modelos para describir el sistema: a) Modelo de objetos b) Modelo dinámico c) Modelo funcional	• Reutilización • Estabilidad • El diseñador piensa en términos del comportamiento de objetos y no en detalles de bajo nivel. • Se construyen clases cada vez más complejas.	• Hay pocos métodos para encontrar inconsistencias en los modelos • Hay pocos métodos para encontrar inconsistencias en los modelos • Al ser un análisis iterativo, es difícil de saber cuándo comenzar con el diseño • Es débil en el diseño.	Aplicable a los lenguajes de programación, al diseño de interfaces de usuario, bases de datos y arquitectura de computadoras.

MODELO DE PROTOTIPOS

En nuestro equipo decidimos, utilizar el modelo de prototipos ya que se nos hace el más cómodo debido a su constate revisión del proyecto a través de iteraciones (visualizaciones preliminares del proyecto), además de tener la ventaja de que nuestro “cliente” puede apreciar el desarrollo más actualizado del proyecto y darnos su punto de vista respecto a los cambios que la aplicación pueda sufrir y darnos sugerencias para su producto, además de que como ventaja es que en un entorno laboral es un modelo que no requiere de un presupuesto elevado y es útil para administrar adecuadamente los tiempos y reducirlos.