Análisis y Diseño I

Matriz comparativa de modelos de desarrollo de software

Tipo de modelo	Enfoque	Nombre	Ventajas	Desventajas	Aplicaciones	Organización mínima requerida	Modelo de proceso
Tradicionales		Ciclo de vida clásico	1) Etapas conocidas.	1) El software terminado se tiene hasta el final de las pruebas, hasta ese momento se hace "tangible" ante el usuario. 2) Se debe tener el debe tenerse claro el análisis del negocio y del sistema.	1) Cuando se tiene claro el análisis del negocio y del sistema y se cuenta con una correcta estimación de tiempos. 2) Cuando hay poco personal al inicio, se tiene que entregar lo más importante del sistema rápido.		
Incremental		Modeo en V	1) Integra el enfoque de aseguramiento de la calidad en cada una de las etapas del ciclo de vida clásico.	1) Para probar la calidad de las etapas iniciales, se debe esperar a que se haya terminado la codificación de todo el sistema.	1) Integrar un enfoque de calidad de manera explícita en ciclo de vida del software.		
Evolutivas	Evolutivas tradicionales	Prototipos	1) Al cliente le gusta ver un sistema funcionando 2)Al desarrollador le gusta empezar a programar	1) El cliente ve lo que parece ser un software, no se sabe la calidad con la que está desarrollado 2) El desarrollador implementa la tecnología conocida.	1) Cuando no se tiene claro el concepto del sistema desde el punto de vista del ingeniero de software/cliente. 2) Como estratega de análisis.		
		Espiral	1) La funcionalidad se va enriqueciendo con cada iteración 2) La primera iteración permite tener el análisis del sistema distribuido en cada eje.	1) Es difícil de convencer a clientes que el enfoque es controlable.	1) Cuando se tiene habilidad de evaluación de riegos.		
	Ágiles	XP	1) Da poder y responsabilidad a los desarrolladores 2)El conocimiento de la arquitectura del software está distribuido 3) El cliente percibe entregas rápidas de software 4) El usuario se involucra más en el desarrollo de software.	1) Dificil implementar en grandes empresas 2) Paradigma de trabajo individual de desarrolladores 3) Difícil estimar costos reales porque se tiene retroalimentación luego de las entregas.	1) Cuando se tiene un cliente disponible y comprometido y se necesita liberar rápidamente versiones de software.		
		SCRUM	1) Liberaciones regulares de software 2) Inspecciones constantes 3) Propiedad individual del código	1) Riesgo en definir una arquitectura no escalable	1) Cuando se pueden identificar las características del software y se necesita liberar rápidamente versiones de software.		
Modelos de referencia		RUP	1) El uso de UML como estándar de facto hace que peque a lo manejado en la industria 2) Las fases se adaptan a la organización.	1) Empresas con sistemas híbridos. No todo encaje en diseño orientado a objetos 2) El conocimiento que demanda en ingenieros de software es más que el promedio.	1) Cuando se van a definir arquitecturas orientadas a objetos y se mezcla al software como producto y como proceso.		
		EUP	1) Extiende RUP a una vista empresarial orientada a portafolios de proyectos 2) Incluye las fases de producción y retiro 3) Puede ser combinado con otras metodologías como XP.	1) Puede ser difícil de implementar si no se tiene la experiencia de la metodología en el negocio.	1) Cuando se requiera agregar a la metodología el alcance de de administración de proyectos. 2) Cuando se quiera retirar un sistema de producción.		
		CMM	1) Beneficios para empresas certificadas 2) Compatible con procedimientos corporativos para la entrega de proyectos.	1) Es caro de implementar y mantener 2) Se debe establecer un comité de crecimiento 3) Requiere mucha documentación 4) Una empresa madura en CMM no garantiza entregas exitosas de software (como producto).	1) Cuando se quiera certificar la madurez de los procesos de las instituciones más allá del software.		
Orientados a diseño de arquitecturas		Zachman Framework	1) Muestra las vistas de una arquitectura 2) Recuerda puntos clave a tomar en cuenta 3) Define los roles presentes en definición de arquitectura	1) Exceso de documentación. 2) Puede ser muy pesado en el desarrollo de software.	1) Cuando no se desea una guía paso a paso en la elaboración de una arquitectura, cuando no se tiene experiencia en diseño de software y se tiene compromiso de usuarios de negocios.		
		Model driven	1) Separación de vistas de la plataforma 2) Estrategia viable para integración de sistemas 3) Promueve la modelación de procesos de negocios	1) La tecnología cambia mucho y hacer mapeos automaticos es difícil 2) Requiere muchos conocimientos de los lenguajes estándar 3) Propone planos teóricos para eruditos que no han escrito una línea de código nunca	1) Cuando se quiere diseñar una arquitectura implementable en cualquier plataforma con estándares de la industria. 2)Para producir modelos.		