

Actividades genéricas de un proceso de software

Modelo de proceso software

Modelos clásicos

Modelos especializados

Métodos ágiles

Constituyen un conjunto de actividades cuya meta es el desarrollo del software (producto). Aunque existen muchos procesos diferentes de software, existen algunas actividades básicas comunes a todos ellos:

- Entender el problema (Ing de requisitos): Estudio de viabilidad, análisis de requisitos, especificación, validación de requisitos
- Pensar una solución (Diseño de software): La estructura del software que se va a implementar, los datos que forman parte del sistema, las interfaces entre los componentes del sistema, los algoritmos utilizado.
- Llevar a cabo un plan (Implementación del software): Producir software de acuerdo a su especificación
- Examinar el resultado y comprobar que el problema se ha resuelto de forma satisfactoria (validación y evolución): El software debe de hacer lo que el cliente desea, debe evolucionar para satisfacer las necesidades cambiantes del cliente

Actividades genéricas de un proceso de software

- Comunicación: Saber que quiere el cliente
- Planificación: descripción de las tareas técnicas a llevar a cabo
- Modelado: Creación de modelos para entender los requisitos software y el diseño que los llevará a cabo
- Construcción: generación de código (programas) y su testeo para eliminar errores (software testing)
- Implantación: entrega al cliente y su puesta en marcha

Existen actividades complementarias como la gestión de proyectos, gestión de riesgos, control de calidad, revisiones técnicas, gestión de configuraciones

Modelo de proceso software

Constituyen una representación abstracta de un proceso software. Debido a la amplia diversidad de tipos de aplicaciones software. No existe un proceso de software ideal, hay modelos distintos que se adaptan al tipo de software. Si el modelo no es adecuado, se presentan dificultades. Un modelo será apropiado si el software resultante es de calidad, el tiempo de entrega se ajusta al previsto y la viabilidad a largo plazo del producto que se construye

Modelos clásicos

Nombre	Descripción	Ventajas	Inconvenientes	Diagrama
Cascada	Secuencial, clásico	Adecuado si los requisitos están totalmente claros	El cliente tiene que tener paciencia y es poco flexible	
Prototipo	Desarrolla un prototipo usando pocos recursos y poco tiempo. El cliente ofrece una retroalimentación antes de la entrega de un producto final. Los requisitos cambian constantemente	Útil si se conoce los objetivos generales pero no los detalles concisos. Se puede reutilizar el prototipo	El prototipo puede ser totalmente desechado por no cumplir los requisitos. Poca documentación	
Espiral	Evolutivo e iterativo. Cada ciclo representa una fase del proyecto. Se centra en prever riesgos. Si el cliente no está satisfecho, repetimos otro ciclo.	Realista, mejor conocimiento del riesgo. Usa prototipos	Requiere mucho tiempo, muy costoso y requiere expertos en gestión de riesgos	
Incremental	Presentar una implementación inicial e ir refinando a través de diferentes versiones	Desarrollo creciente, satisface las necesidades inmediatas	Difícil establecer una arquitectura estable del sistema, inapropiado para sistemas grandes. Los cambios continuos tienden a corromper la estructura del software	
Proceso Unificado de Rational	Combina los métodos clásicos con ágiles. Dividido en ciclos, relacionado con el negocio, cada ciclo corresponde con una versión del sistema	Software completo (código fuente, componentes, manuales, productos asociados)	Mucho papeleo, lento y todo debe ser documentado	

Modelos especializados

Enfoques concretos de ingeniería del software que se aplican en algunos contextos, entre ellos destacan:

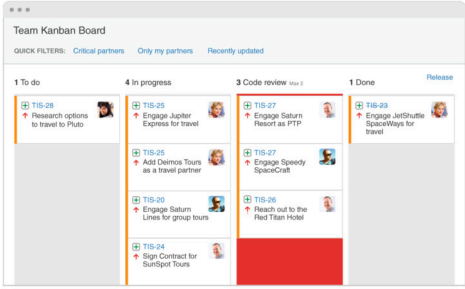
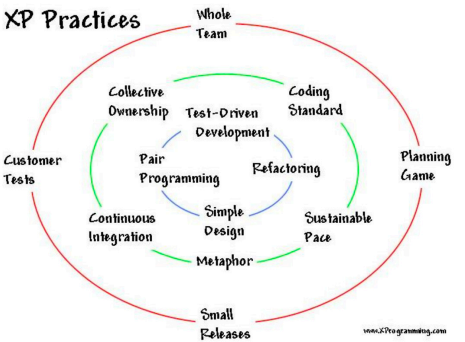

- Desarrollo basado en componentes: Reutilización de código. Un programa se construye integrando componentes débiles acoplados
- Métodos formales: Especificación matemática del software. A través del análisis matemático se detectan errores e inconsistencias

Métodos ágiles

Buscan la satisfacción del cliente basados en un desarrollo incremental. Utilizan métodos informales sencillos y equipos pequeños para desarrollar el software. Las guías del desarrollo se basan en la comunicación continua y activa con el cliente. Se basan en la idea del constante cambio de los requisitos por parte del cliente ofreciendo un método muy adaptable a estos cambios





Nombre	Descripción	Diagrama
Kanban	Visualización, priorización, mejora continua, liderazgo en todos los niveles, calidad garantizada	
Programación externa (XP)	Ideal para software cambiante de alto riesgo. Pequeños equipos, comunicación, simplicidad, retroalimentación y muchos tests	
Scrum	Requisitos inestables que requieren rapidez y flexibilidad. Desarrollo incremental donde solapan las diferentes fases del desarrollo. Proceso iterativo que define un conjunto de prácticas y roles. Útil para proyectos pequeños y medianos	

	Scrum	Kanban
Cadencia	Sprints de longitud fija periódicos	Flujo continuo
Release	Al final de cada sprint, si lo aprueba el propietario del producto	Entrega continua o a discreción del equipo
Funciones	Propietario del producto, experto en scrum, equipo de	No existen funciones. Algunos equipos cuentan con la ayuda de un orientador
Métricas	Velocidad	Tiempo del ciclo
Filosofía de cambios	Los equipos deben evitar cambios en la previsión durante el sprint. De lo contrario, se sacrifica el aprendizaje sobre la estimación.	Los cambios pueden suceder en cualquier momento.