Enfoques de Desarrollo de Software

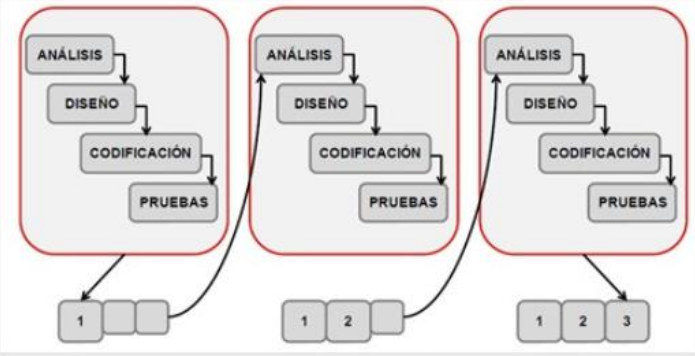
Dentro de los enfoques más generales (clásicos) de desarrollo de software podemos citar:

- Modelo en cascada (1970)
- Prototipado.
- Incremental (Mills;1980)
- Espiral (Boehm;1986)
- RAD-Rapid Application Development(Maslow;1980)

CASCADA

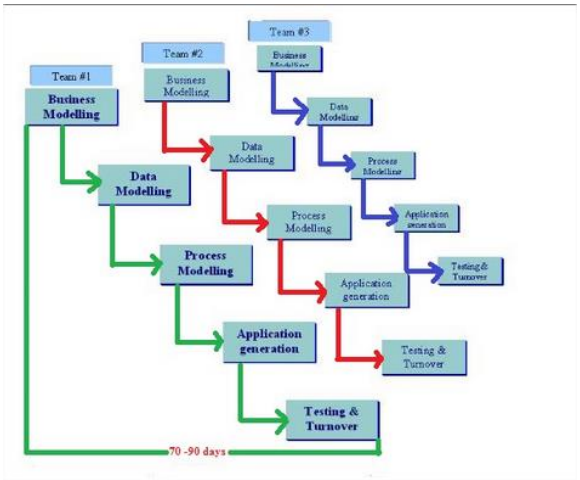
PROTOTIPADO

INCREMENTAL



ESPIRAL

RAD



ENFOQUES	VENTAJAS	DESVENTAJAS
CASCADA	De partida se cuenta con los requerimientos muy completos y consistentes. Disminuye el efecto *"bola de nieve"* al reducir el mantenimiento, considerando que se tienen unas especificaciones completas y correctas.	Si el cliente no es muy claro de lo que exactamente quiere es un problema para este modelo. Cualquier cambio que se menciona en el medio del proceso, puede causar mucha confusión. La mayor desventaja del modelo de cascada es que mientras no se finalice la etapa final del ciclo de desarrollo, el cliente no tendrá algo en sus manos. Por lo tanto, es difícil mencionar si lo que se ha diseñado es exactamente lo que había solicitado.
PROTOTIPADO	No modifica el flujo del ciclo de vida. Reduce el riesgo de construir productos que no satisfagan las necesidades de los usuarios. Reduce costo y aumenta la probabilidad de éxito. Este modelo es útil cuando el cliente conoce los objetivos generales para el software, pero no identifica los requisitos detallados de entrada, procesamiento o salida. También ofrece un mejor enfoque cuando el responsable del desarrollo del software está inseguro de la eficacia de un algoritmo, de la adaptabilidad de un sistema operativo o de la forma que debería tomar la interacción humano-máquina.	Debido a que el usuario ve que el prototipo funciona, piensa que este es el producto terminado y no entienden que recién se va a desarrollar el software. El desarrollador puede caer en la tentación de ampliar el prototipo para construir el sistema final sin tener en cuenta los compromisos de calidad y mantenimiento que tiene con el cliente
INCREMENTAL	Con un paradigma incremental se reduce el tiempo de desarrollo inicial, ya que se implementa la funcionalidad parcial. También provee un impacto ventajoso frente al cliente, que es la entrega temprana de partes operativas del Software. El modelo proporciona todas las ventajas del modelo en cascada realimentado, reduciendo sus desventajas sólo al ámbito de cada incremento. Permite entregar al cliente un producto más rápido en comparación del modelo de cascada. Resulta más sencillo acomodar cambios al acotar el tamaño de los incrementos. Por su versatilidad requiere de una planeación cuidadosa tanto a nivel administrativo como técnico.	El modelo Incremental no es recomendable para casos de sistemas de tiempo real, de alto nivel de seguridad, de procesamiento distribuido, y/o de alto índice de riesgos. Requiere de mucha planeación, tanto administrativa como técnica. Requiere de metas claras para conocer el estado del proyecto.
ESPIRAL	Los desarrolladores pueden describir las características de alta prioridad para luego proceder con su desarrollo a través de la construcción de un prototipo. Este prototipo se prueba y si es óptimo, se incorporan en el nuevo sistema. La adaptabilidad del modelo hace posible su adaptabilidad a cualquier número de cambios que puedan ocurrir durante cualquier fase del proyecto. Dado que la construcción de prototipos se realiza en pequeños fragmentos o trozos, la estimación de costos se transforma en algo mas fácil y el cliente puede obtener el control sobre la administración del nuevo sistema.	El modelo de espiral funciona mejor para los grandes proyectos, donde los costos son mucho más altos y los requisitos del sistema implica un mayor nivel de complejidad. El modelo espiral debe trabajar en un protocolo, que debe ser seguido estrictamente para su buen funcionamiento. A veces se hace difícil seguir dicho protocolo. La evaluación de los riesgos involucrados en el proyecto, pueden disparar el costo y puede ser mayor que el costo de la construcción del sistema.
RAD	Comprar puede ahorrar dinero en comparación con construir. Los entregables pueden ser fácilmente trasladados a otra plataforma. El desarrollo se realiza a un nivel de abstracción mayor. Visibilidad temprana. Mayor flexibilidad. Menor codificación manual. Mayor involucramiento de los usuarios. Posiblemente menos fallas. Posiblemente menor costo. Ciclos de desarrollo más pequeños.	Comprar puede ser más caro que construir. Costo de herramientas integradas y equipo necesario. Menor precisión científica. Más fallas (por síndrome de "codificar a lo bestia"). Prototipos pueden no escalar, un problema mayúsculo.

Una **metodología de desarrollo de software** se refiere a un framework que es usado para estructurar, planear y controlar el proceso de desarrollo en sistemas de información.

Al finalizar los noventa surgen interesantes metodologías para el desarrollo de Software, todas ellas basadas en alguno de los enfoques que se mencionan en este documento (5 enfoques clásicos).

Metodología de Desarrollo de Software

1970

- Programación estructurada sol desde 1969
- Programación estructurada Jackson desde 1975

1980

- Structured Systems Analysis and Design Methodology (SSADM) desde 1980
- Structured Analysis and Design Technique (SADT) desde 1980
- Ingeniería de la información (IE/IEM) desde 1981

1990

- Rapid application development (RAD) desde 1991.
- Programación orientada a objetos (OOP) a lo largo de la década de los 90's
- Virtual finite state machine (VFSM) desde 1990s
- Dynamic Systems Development Method desarrollado en UK desde 1995.
- Scrum (desarrollo), en la última parte de los 90's
- Rational Unified Process (RUP) desde 1999.
- Extreme Programming(XP) desde 1999

Nuevo milenio

- Enterprise Unified Process (EUP) extensiones RUP desde 2002
- Constructionist design methodology (CDM) desde 2004 por Kristinn R. Thórisson
- Agile Unified Process (AUP) desde 2005 por Scott Ambler

Dentro de las metodologías de desarrollo más utilizadas por estos días, podemos reconocer en ellas algunas características.

METODOLOGÍA	CARACTERÍSTICAS
AGILE	Con esta metodología, el desarrollo de software se divide en proyectos más pequeños, por lo que es ideal para los ciclos de lanzamiento corto y finitos, como los requeridos para las actualizaciones de un producto existente. Es conveniente cuando se tiene experiencia en la preparación de documentación, el equipo de desarrollo trabaja estrechamente y, preferiblemente, también en la misma ubicación física.
SCRUM	Los proyectos que utilizan esta metodología otorgan un peso considerable a la inteligencia, experiencia y habilidades que los miembros del equipo de desarrollo aportan a la hora de resolver problemas. Las tareas de proyecto se llevan a cabo en ciclos cortos conocidos como sprints, muy manejables y adecuadamente priorizados, que permiten mostrar el progreso de forma muy sencilla. El proyecto más largo se beneficiaría de la estructuración de éste método, en comparación con otros modelos de desarrollo de software y uno de los motivos es que los desarrolladores se sienten comprometidos con las metas y responsables del éxito de la iniciativa.

PROGRAMACIÓN EXTREMA	Su enfoque de evolución constante resulta muy beneficioso, sobre todo, si se tiene en cuenta que, uno de los procedimientos a aplicar es descomponer las tareas de desarrollo para incluir sólo las actividades críticas. Precisamente por eso resulta un método muy indicado para los casos en que el entorno de desarrollo carece de estabilidad o si los requisitos aplicables al proyecto son inciertos.
PROGRAMACIÓN EN PAREJA	Consiste en juntar a dos programadores para que trabajen juntos en una sola estación de trabajo, facilitando que cada desarrollador se centre en diferentes aspectos del proyecto para conseguir que se reduzcan los costes de las pruebas y también los defectos en los programas. Merece la pena apostar por esta alternativa cuando el equipo de desarrollo es pequeño y seguro, y siempre que los desarrolladores estén de acuerdo.
KANBAN	En base al sistema just in time, esta metodología trata de evitar los cuellos de botella, mediante una identificación muy precisa de las tareas en curso que funciona muy bien en proyectos que no sean demasiado complejos. Permite ser complementado por otras metodologías y es muy recomendable su utilización cuando se tiene confianza plena en el equipo de desarrollo.