Metodologías de Desarrollo de Aplicaciones Móviles

# a) Modelo en cascada. b) Desarrollo rápido de aplicaciones.

# c) Scrum.

# d) Mobile-D. FABIO

# e) Desarrollo de Software Adaptable. FABIO

# f) Prototipos. FABIO

# Modelo en cascada

Es un enfoque metodológico que ordena rigurosamente las etapas del [proceso para el desarrollo de software](http://es.wikipedia.org/wiki/Proceso_para_el_desarrollo_de_software), de tal forma que el inicio de cada etapa debe esperar a la finalización de la etapa anterior. Al final de cada etapa, el modelo está diseñado para llevar a cabo una revisión final, que se encarga de determinar si el proyecto está listo para avanzar a la siguiente fase.

Este modelo es solo aplicable cuando están totalmente cerrados los requisitos y no van a cambiar. No hay retroalimentación entre las fases en que se divide el proyecto. Por lo que cada fase se va cerrando de forma secuencial. Todo el proceso está fijado por fechas límites y presupuestos. Este modelo sólo es aconsejable para proyectos móviles muy controlados y previsibles. Las fases son Análisis de requisitos, Diseño del Sistema, Diseño del Programa, Codificación, Pruebas, Verificación, Mantenimiento.

Los principios básicos del modelo de cascada según (Winston, 2007) son los siguientes:

* El proyecto está dividido en fases secuenciales, con cierta superposición y splashback aceptable entre fases.
* Se hace hincapié en la planificación, los horarios, fechas, presupuestos y ejecución de todo un sistema de una sola vez.
* Un estricto control se mantiene durante la vida del proyecto a través de la utilización de una amplia documentación escrita, así como a través de comentarios y aprobación / signoff por el usuario y la tecnología de la información de gestión al final de la mayoría de las fases antes de comenzar la próxima fase.

## 

## Ventajas

* Realiza un buen funcionamiento en equipos débiles y productos maduros, por lo que se requiere de menos capital y herramientas para hacerlo funcionar de manera óptima.
* Es un modelo fácil de implementar y entender.
* Está orientado a documentos.
* Es un modelo conocido y utilizado con frecuencia.
* Promueve una metodología de trabajo efectiva: Definir antes que diseñar, diseñar antes que codificar.

## Desventajas

* En la vida real, un proyecto rara vez sigue una secuencia lineal, esto crea una mala implementación del modelo, lo cual hace que lo lleve al fracaso.
* El proceso de creación del software tarda mucho tiempo ya que debe pasar por el proceso de prueba y hasta que el software no esté completo no se opera. Esto es la base para que funcione bien.
* Cualquier error de diseño detectado en la etapa de prueba conduce necesariamente al rediseño y nueva programación del código afectado, aumentando los costos del desarrollo.
* Una etapa determinada del proyecto no se puede llevar a cabo a menos de que se haya culminado la etapa anterior.

# Desarrollo rápido de aplicaciones

El desarrollo rápido de aplicaciones (RAD) es una metodología de desarrollo de software introducido por (James, 2013), que implica el desarrollo iterativo y la construcción de prototipos. El desarrollo rápido de aplicaciones es un término originalmente utilizado para describir un [proceso de desarrollo de software](https://es.wikipedia.org/wiki/Proceso_de_desarrollo_de_software).

Se da énfasis a la obtención de un prototipo funcional de una aplicación para posteriormente ir mejorándolo incluyendo más funcionalidades y complejidad. Es recomendable el uso de patrones de diseño bien conocidos para adaptarse a los cambios de requisitos.

Se suele usar cuando los plazos de entrega son muy cortos y se precisa tener un entregable de forma inmediata. No se descarta utilizar otras metodologías de forma posterior, ya que este tipo de desarrollo puede ser usado para mostrar un esbozo de la aplicación a un cliente, generalmente en un par de días. El Desarrollo rápido de aplicaciones consiste en un ciclo de desarrollo corto basado en tres fases (Requisitos, Diseño y Construcción) con un plazo de entrega ideal de 90 a 120 días como máximo con equipos de aproximadamente 6 personas.

Los principios básicos del desarrollo rápido de aplicaciones según (James, 2013) son :

* El objetivo clave es para un rápido desarrollo y entrega de una alta calidad en un sistema de costo relativamente bajo de inversión.
* Intenta reducir los riesgos inherentes del proyecto partiéndolo en segmentos más pequeños y proporcionar más facilidad de cambio durante el proceso de desarrollo.
* Orientación dedicada a producir sistemas de alta calidad con rapidez, principalmente mediante el uso de iteración por prototipos (en cualquier etapa de desarrollo), promueve la participación de los usuarios y el uso de herramientas de desarrollo computarizadas. Estas herramientas pueden incluir constructores de [Interfaz gráfica de usuario](https://es.wikipedia.org/wiki/Interfaz_gr%C3%A1fica_de_usuario) (GUI), [Computer Aided Software Engineering](https://es.wikipedia.org/w/index.php?title=Computer_Aided_Software_Engineering&action=edit&redlink=1" \o "Computer Aided Software Engineering (aún no redactado)) (CASE) las herramientas, los [sistemas de gestión de bases de datos](https://es.wikipedia.org/wiki/Sistemas_de_gesti%C3%B3n_de_bases_de_datos) (DBMS), lenguajes de programación de cuarta generación, generadores de código, y técnicas orientada a objetos.
* Hace especial hincapié en el cumplimiento de la necesidad comercial, mientras que la ingeniería tecnológica o la excelencia es de menor importancia.
* Control de proyecto implica el desarrollo de prioridades y la definición de los plazos de entrega. Si el proyecto empieza a aplazarse, se hace hincapié en la reducción de requisitos para el ajuste, no en el aumento de la fecha límite.
* En general incluye [Joint application development](https://es.wikipedia.org/wiki/Joint_application_development" \o "Joint application development) (JAD), donde los usuarios están intensamente participando en el diseño del sistema, ya sea a través de la creación de consenso estructurado en talleres, o por vía electrónica.
* La participación activa de los usuarios es imprescindible.
* Iterativamente realiza la producción de software, en lugar de enfocarse en un prototipo.
* Produce la documentación necesaria para facilitar el futuro desarrollo y mantenimiento.

## Ventajas:

* Enfatiza ciclos de desarrollo extremadamente cortos.
* Tiene las ventajas del modelo clásico.
* Se asegura de que el producto entregado cumple las necesidades del cliente.
* El desarrollo se realiza a un nivel de abstracción mayor.
* Visibilidad temprana.
* Mayor involucramiento de los usuarios.
* Posiblemente menos fallas.
* Posiblemente menor costo.
* Ciclos de desarrollo más pequeños

## Desventajas:

* Solo se puede aplicar si el sistema se puede modularizar de forma que permita completarse cada una de las funciones principales en menos de tres meses.
* Para proyectos grandes puede requerir muchos equipos de trabajo distintos.
* Requiere clientes y desarrolladores comprometidos en las rápidas actividades necesarias.
* No resulta adecuado cuando los riesgos técnicos son elevados.
* Se pueden tener problemas con la aceptación del prototipo.
* Costo de herramientas integradas y equipo necesario.
* Progreso más difícil de medir.
* Menor precisión científica

# Scrum

Scrum se centra la gestión de proyectos en situaciones en las que es difícil planificar el futuro, con mecanismos de control “proceso empírico”, donde los bucles de realimentación constituyen el elemento central. El software es desarrollado por un equipo de auto-organización en incrementos (llamados “sprints”), empezando por la planificación y finalizando con un comentario. Las características que deben aplicarse en el sistema se registran en un backlog. Entonces, el dueño del producto decide qué elementos del backlog se deben desarrollar en el sprint siguiente. Los miembros del equipo coordinan su trabajo en un diario de stand-up de la reunión. Un miembro del equipo, el “Scrum Master” (equivalente al gerente del proyecto), es el encargado de resolver los problemas que impiden que el equipo trabaje eficazmente, este equipo generalmente es de diez o menos componentes. Las fases son

Los principios de Scrum propuestos por (Schwaber, 1995) son:

* Los individuos por encima de los procesos y herramientas:debe de haber una cohesión de equipo fuerte, porque el triunfo de un hito no es el triunfo de un solo jugador sino de todo el equipo, él mismo entrega el resultado. Todos colaboran para obtener el triunfo y empujan al que no está caminando como se debe.
* Se entrega soluciones por encima de reportes de seguimiento: Se centra en presentar al cliente la solución que él pueda operar y usar, no solamente en entregar un reporte de lo que se ha hecho, de esta forma el cliente ve el progreso y puede decir cuando o no parar. Esto es una fortaleza ya que la mayoría está acostumbrada a un plan y el resultado lo ve al final del proyecto.
* Adaptarse a los cambios en lugar de señirse a un plan especifico: el cliente va viendo el resultado del producto y decide si sigue o termina el producto en ese momento. O inclusive tan radical como se escucha darle un giro completo.

## Ventajas

* Se obtiene software lo más rápido posible y este cumple con los requerimientos más importantes.
* Se trabaja en iteraciones cortas, de alto enfoque y total transparencia.
* Se acepta que el cambio es una constante universal y se adapta el desarrollo para integrar los cambios que son importantes.
* Se incentiva la creatividad de los desarrolladores haciendo que el equipo sea auto administrado.
* Se mantiene la efectividad del equipo habilitando y protegiendo un entorno libre de interrupciones e interferencias.
* Permite producir software de una forma consistente, sostenida y competitiva.
* Las reuniones se dedican a inconvenientes recientes, evitando el estancamiento

## Desventajas

* Si no existe una fecha definitiva de finalización del proyecto es posible que se siga solicitando, y añadiendo, nueva funcionalidad.
* Si una tarea no esta bien definida, los costes de tiempo y dinero estimados del proyecto no serán demasiado exactos. En ese caso, la tarea se puede extender sobre varios sprints.
* Si los miembros del equipo no están centrados y convencidos, el proyecto nunca se completara o incluso fallará.
* Esta metodología necesita solo miembros de equipo experimentados. Si el equipo consiste en gente que son junior, el proyecto no puede ser completado a tiempo.
* Además de los recursos sin suficiente experiencia , la falta de dirección firme pueden llevar a los proyectos a no completarse o incluso fallar.
* La metodología Scrum funciona bien cuando el scrum master confía en el equipo que lleva. Si se practican controles muy estrictos sobre los miembros del equipo, puede ser extremadamente frustrante para ellos, llevando a la desmoralización y el fallo del proyecto.
* Si algunos de los miembros del equipo se marcha durante el desarrollo puede tener un efecto negativo enorme en el desarrollo del proyecto.
* El control de la calidad del proyecto es difícil de implementar y cuantificar a menos que el equipo de test puedan llevar a cabo testeo de regresión después de cada sprint.

# Mobile-D

El objetivo de este método es conseguir ciclos de desarrollo muy rápidos en equipos muy pequeños. Se compone de distintas fases: exploración, inicialización, fase de producto, fase de estabilización y la fase de pruebas. Cada una tiene un día de planificación y otro de entrega.

En la fase de exploración se centra la atención en la planificación y a los conceptos básicos del proyecto. Aquí es donde hacemos una definición del alcance del proyecto y su establecimiento con las funcionalidades donde queremos llegar.

En la iniciación configuramos el proyecto identificando y preparando todos los recursos necesarios como hemos comentado anteriormente en esta fase la dedicaremos un día a la planificación y el resto al trabajo y publicación.

En la fase de producto se repiten interativamente las subfases. Se usa el desarrollo dirigido por pruebas (TDD), antes de iniciar el desarrollo de una funcionalidad debe existir una pueba que verifique su funcionamiento. En esta fase podemos decir que se lleva a acabo toda la implementación.

Después de la fase de producto llega la fase de estabilización en la que se realizan las acciones de integración para enganchar los posibles módulos separados en una única aplicación.

Fase de pruebas. Una vez parado totalmente el desarrollo se pasa una fase de testeo hasta llegar a una versión estable según lo establecido en las primeras fases por el cliente. Si es necesario se reparan los errores, pero no se desarrolla nada nuevo.

Una vez acabada todas las fases deberíamos tener una aplicación publicable y entregable al cliente.

<http://www.genbetadev.com/desarrollo-aplicaciones-moviles/metodos-aplicables-para-el-desarrollo-de-aplicaciones-moviles>

Ventajas:

* Posee bajos costos al realizar un cambio en el proyecto.
* Permite una realización rápida del proyecto.
* Asegura el software adecuado en el momento adecuado.

Desventajas:

* No sirve para grupos de desarrollo demasiado grandes o segmentados.
* Depende demasiado de que exista una buena comunicación entre los miembros del equipo.
* Se deben tener claros los requerimientos

# Desarrollo de Software Adaptable

El método ágil ASD (Adaptive Software Development) es un modelo de implementación de patrones ágiles para desarrollo de software. Al igual que otras metodologías ágiles, su funcionamiento es cíclico y reconoce que en cada iteración se producirán cambios e incluso errores.

El desarrollo de software adaptable (**Adaptive Software Development - ASD**) es una metodología de desarrollo que hace énfasis en aplicar las ideas que se originaron en el mundo de los sistemas complejos, adaptación continua del proceso al trabajo.

**Las principales características del ASD son:**

* Iterativo.
* Orientado a los componentes de software (la funcionalidad que el producto va a tener, características, etc.) más que a las tareas en las que se va a alcanzar dicho objetivo.
* Tolerante a los cambios.
* Guiado por los riesgos
* La revisión de los componentes sirve para aprender de los errores y volver a iniciar el ciclo de desarrollo

ASD utiliza un "cambio orientado hacia el ciclo de vida", que tiene tres componentes que son: especular colaborar y aprender.

**Especular**  
Una primera fase de iniciación para establecer los principales objetivos y metas del proyecto en su conjunto y comprender las limitaciones (zonas de riesgo) con las que operará el proyecto.  
  
En ASD se realizan estimaciones de tiempo sabiendo que pueden sufrir desviaciones. Sin embargo, estas son necesarias para la correcta atención de los trabajadores que se mueven dentro de plazos de forma que puedan priorizar sus tareas.  
  
Se decide el número de iteraciones para consumir el proyecto, prestando atención a las características que pueden ser utilizadas por el cliente al final de la iteración. Son por tanto necesarios, marcar objetivos prioritarios dentro de las mismas iteraciones.  
  
Estos pasos se puede volver a examinar varias veces antes de que el equipo y los clientes están satisfechos con el resultado.  
  
**Colaborar**  
Es la fase donde se centra la mayor parte del desarrollo manteniendo una componente cíclica. Un trabajo importante es la coordinación que asegure que lo aprendido por un equipo se transmite al resto y no tenga que volver a ser aprendido por los otros equipos.  
  
**Aprender**  
La última etapa termina con una serie de ciclos de colaboración, su trabajo consiste en capturar lo que se ha aprendido, tanto positivo como negativo. Es un elemento crítico para la eficacia de los equipos.

**Ventajas**

* La tercera fase del ciclo de vida, revisión de los componentes, sirve para aprender de los errores y volver a iniciar el ciclo de desarrollo.
* Apunta hacia el [Rapid Application Development](http://en.wikipedia.org/wiki/Rapid_application_development) ([RAD)](http://es.wikipedia.org/wiki/Desarrollo_r%C3%A1pido_de_aplicaciones),  el cual enfatiza velocidad de desarrollo para crear un producto de alta calidad, bajo mantenimiento involucrando al usuario lo más posible.
* Utiliza información disponible acerca de cambios para mejorar el comportamiento del software.
* Promulga colaboración, la interacción de personas.
* Anticipa cambios y trata automáticamente con ellos dentro de un programa en ejecución, sin la necesidad de un programador.

**Desventajas**

* Aunque el ciclo entre el aprendizaje y la especulación es bueno permitiéndonos entregar productos con alta calidad, la prolongación de dicho ciclo por errores o cambios que no son detectados en reuniones anteriores afecta tanto a la calidad del producto como a su costo total.
* Dado a que es una metodología ágil implica no realizar procesos que son requeridos en las metodologías tradicionales o por lo menos no realizarlos en procesos diferentes, lo cual implica que empresas grandes las cuales necesitan llevar un mayor control a procesos y personas, tener tareas asignadas a un estado o proceso especifico, y en las cuales dicho incremento de procesos no afectan en gran medida al costo final del producto, para dichas empresas el elegir una metodología tradicional resulta mucho mas rentable tanto por el gran volumen de personal, de productos, y de costos que se manejan y para los cuales se tendrá un mayor control.

# Modelo de Construcción de Prototipos

Los modelos evolutivos son iterativos; los caracteriza la forma en que permiten que los ingenieros de software desarrollen versiones cada vez más completas del software. El diseño rápido se basa en una representación de aquellos aspectos del software que serán visibles para el cliente o el usuario final (por ejemplo, la configuración de la interfaz con el usuario y el formato de los despliegues de salida). El diseño rápido conduce a la construcción de un prototipo, el cual es evaluado por el cliente o el usuario para una retroalimentación; gracias a ésta se refinan los requisitos del software que se desarrollará. La iteración ocurre cuando el prototipo se ajusta para satisfacer las necesidades del cliente. Esto permite que al mismo tiempo el desarrollador entienda mejor lo que se debe hacer y el cliente vea resultados a corto plazo.

**Etapas**

1. Plan rápido
2. Modelado, diseño rápido
3. Construcción del Prototipo
4. Desarrollo, entrega y retroalimentación
5. Comunicación

El paradigma de construcción del prototipo puede ser conducido con uno o dos objetivos en mente:

1. El propósito del prototipado es establecer un conjunto de requerimientos formales que pueden luego ser traducidos en la producción de programas mediante el uso de métodos y técnicas de ingeniería de programación,
2. El propósito de la construcción del prototipo es suministrar un continuo que pueda conducir al desarrollo evolutivo de la producción del software. Ambos métodos tienen sus méritos y ambos crean problemas.

**Ventajas**

* 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.
* No modifica el flujo del ciclo de vida.
* Reduce el riesgo de construir productos que no satisfagan las necesidades de los usuarios.
* Reduce costos y aumenta la probabilidad de éxito.
* Exige disponer de las herramientas adecuadas.
* No presenta calidad ni robustez.
* Una vez identificados todos los requisitos mediante el prototipo, se construye el producto de ingeniería.

**Desventajas**

* El usuario tiende a crearse unas expectativas cuando ve el prototipo de cara al sistema final. A causa de la intención de crear un prototipo de forma rápida, se suelen desatender aspectos importantes, tales como la calidad y el mantenimiento a largo plazo, lo que obliga en la mayor parte de los casos a reconstruirlo una vez que el prototipo ha cumplido su función. Es frecuente que el usuario se muestre reacio a ello y pida que sobre ese prototipo se construya el sistema final, lo que lo convertiría en un prototipo evolutivo, pero partiendo de un estado poco recomendado.
* En aras de desarrollar rápidamente el prototipo, el desarrollador suele tomar algunas decisiones de implementación poco convenientes (por ejemplo, elegir un lenguaje de programación incorrecto porque proporcione un desarrollo más rápido). Con el paso del tiempo, el desarrollador puede olvidarse de la razón que le llevó a tomar tales decisiones, con lo que se corre el riesgo de que dichas elecciones pasen a formar parte del sistema final

Bibliografía

<http://www.genbetadev.com/desarrollo-aplicaciones-moviles/metodos-aplicables-para-el-desarrollo-de-aplicaciones-moviles>

<https://www.exabyteinformatica.com/uoc/Informatica/Tecnologia_y_desarrollo_en_dispositivos_moviles/Tecnologia_y_desarrollo_en_dispositivos_moviles_(Modulo_4).pdf>

<http://congreso.investiga.fca.unam.mx/es/docs/anteriores/xviii/docs/8.01.pdf>

<http://www.researchgate.net/publication/265207670_Propuesta_Metodolgica_para_Desarrollo_de_Aplicaciones_Mviles_para_Dispositivos_Android>

<https://www.imaginanet.com/scrum-es-una-metodologia-para-la-programacion-de-aplicaciones-moviles-y-web.html>

<http://www.uelbosque.edu.co/sites/default/files/publicaciones/revistas/revista_tecnologia/volumen12_numero2/12Articulo_Rev-Tec-Num-2.pdf>

<http://www.adamwesterski.com/wp-content/files/docsCursos/Agile_doc_TemasAnv.pdf>

<http://openaccess.uoc.edu/webapps/o2/bitstream/10609/17885/1/mtrigasTFC0612memoria.pdf>

<http://congreso.investiga.fca.unam.mx/es/docs/anteriores/xviii/docs/8.01.pdf>

<https://www.exabyteinformatica.com/uoc/Informatica/Tecnologia_y_desarrollo_en_dispositivos_moviles/Tecnologia_y_desarrollo_en_dispositivos_moviles_(Modulo_4).pdf>

[https://www.ingenieriade**software**.mex.tl/images/18149/informe.doc](https://www.ingenieriadesoftware.mex.tl/images/18149/informe.doc)

<https://www.scribd.com/doc/43653369/Modelo-de-Construccion-de-Prototipos>

http://www.monografias.com/trabajos12/proto/proto.shtml