# Universidad Cenfotec

# 60_gr_LOGO_blanco

# Investigación de metodologías de desarrollo del *software*.

# logo

# Profesores:

# Álvaro Cordero

# Pablo Monestel

# Fecha de entrega: 24/06/2015

# Segundo cuatrimestre 2015

**Investigación de metodologías de desarrollo.**

# Introducción.

Con el presente documento se pretende tener una visión más clara acerca de los aspectos básicos de las metodologías de desarrollo. Se abordan dos metodologías de desarrollo: *Iconix* como metodología tradicional y *Kanban* como metodología ágil; estos temas se explicarán ampliamente a lo largo del presente documento.

Con este documento se pretende dar explicación a dos corrientes distintas para el desarrollo de proyectos de software. Se resaltarán ventajas y desventajas de ambas. Además, cuales son las implicaciones de cada una de estas metodologías y la manera de implementarlas.

# ¿Qué son las metodologías de desarrollo?

Según varias definiciones, podemos describir las metodologías de desarrollo como los marcos de trabajo para estructurar, controlar y planificar el proceso de desarrollo en la industria del *software*. Es decir, es una manera de mantener control en el proceso de desarrollo. Lo podemos reducir a un *framework* para estructurar el trabajo de desarrollo.

A lo largo de la historia se han dado varias propuestas en el área de las metodologías de desarrollo. Algunas de estas propuestas se diferencian por su fortaleza, esto las hace más populares que otras y de más rápida implementación, lo que hace que las personas se acoplen a su sistema de trabajo.

Una determinada metodología no es necesariamente aplicable a todo tipo de proyecto, por el contrario, un tipo de proyecto tiene alguna metodología a la que se adapta mejor.

Una metodología de desarrollo de *software* consiste en:

* Una filosofía de desarrollo de *software,* con una base de procesos también en el área de desarrollo de *software.*
* Tener múltiples herramientas, modelos y métodos para asistir en el proceso de desarrollo de *software*.
* Estar documentada con alguna clase de documentación formal.
* Ser promovida por alguna organización, ya sea pública o privada. Esta organización es la que se encarga de promover la metodología.

# Metodologías tradicionales.

Teniendo en cuenta la filosofía de desarrollo de las metodologías, aquellas con mayor énfasis en la planificación, control del proyecto y en la especificación precisa de requisitos y modelado, reciben el apelativo de metodologías tradicionales o pesadas.

Estas metodologías tradicionales imponen una disciplina de trabajo sobre el proceso de desarrollo del *software*, con el fin de conseguir un producto más eficiente. Para ello se hace énfasis en la planificación total del trabajo a realizar y una vez que está detallado comienza el ciclo de desarrollo del producto *software*. Se centran especialmente en el control del proceso mediante una rigurosa definición de roles, actividades, artefactos, herramientas y notaciones para el modelado y documentación detallada. Además, las metodologías tradicionales no se adaptan adecuadamente a los cambios, por lo que no son métodos adecuados cuando se trabaja en un entorno donde los requisitos no pueden predecirse o bien pueden variar (Acuña, 2009).

Cada metodología de desarrollo tiene más o menos su propio enfoque de en lo que debería consistir un proyecto de desarrollo de *software*, pero todas ellas se basan en una serie de enfoques generalistas como lo son:

* *Waterfall Model* – Lineal.
* *Prototyping* – Iterativo.
* *Incremental* - Iterativo y lineal.
* *Spiral* - Iterativo y lineal.
* *Rapid Application Development* (RAD) – Iterativo.

# Metodologías ágiles.

Las metodologías ágiles son una serie de técnicas para la gestión de proyectos que han surgido en contraposición a los métodos clásicos de gestión. Aunque surgieron en el ámbito del desarrollo de *software*, también han sido exportadas a otro tipo de proyectos.

Los procesos ágiles son una buena elección cuando se trabaja con requisitos desconocidos o variables. Si no existen requisitos estables no existe una gran posibilidad de tener un diseño estable y de seguir un proceso totalmente planificado que no vaya a cambiar ni en tiempo, ni en dinero. En estas situaciones un proceso adaptativo será mucho más efectivo que un proceso predictivo. Por otra parte, los procesos de desarrollo adaptativos también facilitan la generación rápida de prototipos y de versiones previas a la entrega final, lo cual agradará al cliente.

Las metodologías ágiles proporcionan una serie de pautas y principios junto a técnicas pragmáticas que puede que no curen todos los males pero harán la entrega del proyecto menos complicada y más satisfactoria tanto para los clientes como para los equipos de entrega (Acuña, 2009).

Algunas metodologías ágiles son:

* ASD (*Adaptive Software Development*).
* Scrum.
* XP (*Extreme Programming*).
* DSDM (*Dynamic Systems Development Method*).

# Metodologías a desarrollar.

## *ICONIX*.

Es una metodología pesada-ligera de desarrollo del *software* que se halla a medio camino entre un RUP (*Rational Unified Process*) y un XP (*eXtreme Programming*).

*Iconix* deriva directamente del RUP y su fundamento es el hecho de que un 80% de los casos pueden ser resueltos tan solo con un uso del 20% del UML, con lo cual se simplifica muchísimo el proceso sin perder documentación al dejar solo aquello que es necesario. Esto implica un uso dinámico del UML de tal forma que siempre se pueden utilizar otros diagramas además de los ya estipulados si se cree conveniente. *Iconix* se guía a través de casos de uso y sigue un ciclo de vida iterativo e incremental. El objetivo es que a partir de los casos de uso se obtenga el sistema final.

Fue elaborado por *Doug Rosenberg*, y *Kendall Scott* a partir de una síntesis del proceso unificado de los “3 amigos” *Booch*, *Rumbaugh* y *Jacobson*. Es una metodología de desarrollo de software basada en UML.

**Características Principales.**

*ICONIX* cuenta con tres características fundamentales:

* Iterativo e incremental: durante el desarrollo del modelo del dominio y la definición de los casos de uso se producen varias iteraciones. El ciclo de vida incremental consiste en desarrollar por partes el producto de manera que puedas integrarlas funcionalmente y el ciclo de vida iterativo en que cada ciclo de iteración se revisa y mejora el producto.
* El desarrollo se organiza en series de mini-proyectos cortos llamados iteraciones.
* Trazabilidad: cada paso que se realiza está definido por un requisito. Se define la trazabilidad como la capacidad de seguir una relación entre los diferentes artefactos de *software* producidos.
* Dinámica del UML: ofrece un uso dinámico del UML porque utiliza algunos diagramas sin exigir la utilización de todos, como en el caso de RUP (*Rational Unified Process*).

**Fases de *ICONIX.***

* Revisión de los requisitos / análisis de requisitos.
  + Analizar todos los requisitos que formarán parte del sistema y con estos construir el diagrama de clases.
  + Para esta fase se utilizan 3 herramientas:
    - Modelo de dominio: identificar objetos y cosas del mundo real que intervienen con nuestro sistema.
    - Modelo de casos de uso: comportamiento que un usuario realiza dentro del sistema. Comprende de actores, casos de uso y el sistema.
    - Prototipo de interfaz de usuario: implica la creación de un modelo o modelos operativos del trabajo de un sistema.
* Revisión del diseño preliminar / análisis y diseño preliminar.
  + A partir de cada caso de uso se obtendrá una ficha de caso de uso, está formada por un nombre, una descripción, una precondición que debe cumplir antes de iniciarse y una pos-condición que debe cumplir al terminar si termina correctamente.
* Revisión crítica del diseño / diseño.
  + Se registran todos los elementos que forman parte de nuestro sistema.
  + Diagramas de secuencia.
* Implementación.
  + Fase en la que se crea el *software* que será entregado al cliente.

**Ventajas de *ICONIX.***

* Dedicada a la construcción de sistemas de gestión de pequeña y mediana complejidad con la participación de los usuarios finales.
* Los usuarios se hacen participantes más activos en los desarrollos del sistema. Suelen mostrarse más interesados en los prototipos de trabajo que en las especificaciones de diseño.
* La probabilidad de que los usuarios aprueben un diseño y luego rechacen su implementación se reducirá notablemente.

## Kanban.

Se define como “un sistema de producción altamente efectivo y eficiente”, ha contribuido a generar un panorama manufacturero óptimo y competitivo. El origen de la metodología *Kanban* debe buscarse en los procesos de producción *just-in-time* (JIT) ideados por Toyota, en los que se utilizaban tarjetas para identificar necesidades de material en la cadena de producción.

El término *Kanban* ha pasado a formar parte de las llamadas metodologías ágiles, cuyo objetivo es gestionar de manera general cómo se van completando las tareas. *Kanban* es una palabra japonesa que significa “tarjetas visuales”, donde *Kan* es “visual”, y *Ban* corresponde a “tarjeta”.

**Los principios de la metodología *Kanban.***

*Kanban* se basa en una serie de principios que la diferencian del resto de metodologías conocidas como ágiles:

* Comience con lo que hace ahora.
  + Se inicia con las funciones y procesos que ya se tienen y estimula cambios continuos, incrementales y evolutivos a su sistema.
* Se acuerda perseguir el cambio incremental y evolutivo.
  + El método *Kanban* anima a los pequeños y continuos cambios incrementales y evolutivos a su sistema actual.
* Respetar el proceso actual, los roles, las responsabilidades y los cargos.
* Liderazgo en todos los niveles.
  + Se debe alentar hechos de liderazgo en todos los niveles de la organización de los contribuyentes individuales a la alta dirección.

**Las tres reglas de *Kanban***

* Mostrar el proceso.
* Limitar el trabajo en curso.
* Optimizar el flujo de trabajo.

**Mostrar el proceso.**

Consiste en la visualización de todo el proceso de desarrollo mediante un tablero físico que en general es públicamente asequible.

Ejemplo de tablero:



**Limitar el trabajo en curso.**

Consisten en acordar anticipadamente la cantidad de ítems que pueden abordarse por cada proceso, es decir, por columnas del tablero. El principal objetivo de establecer estos límites es detectar cuellos de botella.

Es un valor a tener en cuenta que la resolución de cuellos de botella la mayoría de las veces motiva la colaboración del equipo entre los diferentes procesos.

**Optimizar el flujo de trabajo.**

Imploca que el objetivo una la producción estable, continua y previsible; midiendo el tiempo que el ciclo completo de ejecución del proyecto demanda. Por ejemplo: cantidad de días desde el inicio del análisis hasta el fin del *deploy,* según el ejemplo del tablero anterior se obtiene el *CycleTime*.

Al dividir el CycleTime por el WIP se obtiene el "rendimiento de trabajo", denominado *Throughput*, es decir, la cantidad de ítems que un equipo puede terminar en un determinado período de tiempo.

**Ventajas de *Kanban*.**

* Muy fácil de utilizar, actualizar y asumir por parte del equipo.
* Destaca por ser una técnica de gestión de las tareas muy visual, esto permite ver a golpe de vista el estado de los proyecto.

# Referencias

* Acuña, K. B. (25 de 08 de 2009).

http://www.uma.es/.

Recuperado el 14 de 08 de2013, de http://www.eumed.net/: <http://www.eumed.net/librosgratis/2009c/584/Metodologias%20tradicionales%20y> %20metodologias%20agiles.htm

* https://es.wikipedia.org/wiki/Metodolog%C3%ADa\_de\_desarrollo\_de\_software

Wikipedia, Metodología de desarrollo de software, 25 may 2015.

* <http://comunidad.iebschool.com/iebs/general/metodologia-kanban/>

Laia Gilibets,Qué es la metodología Kanban y cómo utilizarla, 31 julio, 2013.

* <http://www.desarrolloweb.com/articulos/desarrollo-agil-kanban.html> Eugenia Bahit, Desarrollo Ágil con Kanban, 08 de noviembre de 2011.
* http://metodologiaiconix.blogspot.com/ Sater JM ,Metodología ICONIX, 16 de febrero de 2014.