METODOLOGÍAS TRADICIONALES VS. METODOLOGÍAS ÁGILES

Roberth G. Figueroa[1], Camilo J. Solís[2]
Armando A. Cabrera[3]
Universidad Técnica Particular de Loja, Escuela de Ciencias en Computación

Resumen — *Desarrollar un buen software depende de un sinnúmero de actividades y etapas, donde el impacto de elegir la mejor metodología para un equipo, en un determinado proyecto es trascendental para el éxito del producto. El papel preponderante de las metodologías es sin duda esencial en un proyecto y en el paso inicial, que debe encajar en el equipo, guiar y organizar actividades que conlleven a las metas trazadas en el grupo.*
En el presente trabajo se detallan los dos grandes enfoques, tanto metodologías tradicionales y metodologías ágiles, las primeras están pensadas para el uso exhaustivo de documentación durante todo el ciclo del proyecto mientras que las segundas ponen vital importancia en la capacidad de respuesta a los cambios, la confianza en las habilidades del equipo y al mantener una buena relación con el cliente. Se verán diferencias, ventajas, desventajas y cual puede encajar en un proyecto de software, es por ello que, ofrecemos una guía dejando libertad de elección para el lector el poder juzgar y elegir la mejor metodología que se adapte a su equipo de desarrollo.

Palabras Claves — *Metodología, RUP, MSF AUP, Scrum, Metodología Tradicional, Metodología Ágil*

INTRODUCCIÓN

Dentro del desarrollo de software y a la altiva necesidad de que los proyectos lleguen al éxito y obtener un producto de gran valor para nuestros clientes, generan grandes cambios en las metodologías adoptadas por los equipos para cumplir sus objetivos, puesto que, unas se adaptan mejor que otras, al contexto del proyecto brindando mejores ventajas.

Es por eso de la importancia de una metodología robusta que ajustada en un equipo cumpla con sus metas, y satisfaga mas allá de las necesidades definidas al inicio del proyecto.

El éxito del producto depende en gran parte de la metodología escogida por el equipo, ya sea tradicional o ágil, donde los equipos maximicen su potencial, aumenten la calidad del producto con los recursos y tiempos establecidos.

METODOLOGÍA TRADICIONAL

Al inicio el desarrollo de software era artesanal en su totalidad, la fuerte necesidad de mejorar el proceso y llevar los proyectos a la meta deseada, tuvieron que importarse la concepción y fundamentos de metodologías existentes en otras áreas y adaptarlas al desarrollo de software. Esta nueva etapa de adaptación contenía el desarrollo dividido en etapas de manera secuencial que de algo mejoraba la necesidad latente en el campo del software.

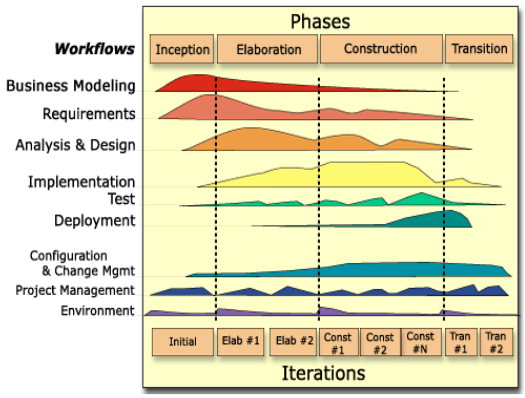
Entre las principales metodologías tradicionales tenemos los ya tan conocidos RUP y MSF entre otros, que centran su atención en llevar una documentación exhaustiva de todo el proyecto y centran su atención en cumplir con un plan de proyecto, definido todo esto, en la fase inicial del desarrollo del proyecto.

Otra de las características importantes dentro de este enfoque tenemos los altos costos al implementar un cambio y al no ofrecer una buena solución para proyectos donde el entorno es volátil.

Las metodologías tradicionales (formales) se focalizan en documentación, planificación y procesos. (Plantillas, técnicas de administración, revisiones, etc.), a continuación se detalla RUP uno de los métodos más usados dentro de los métodos tradicionales

RATIONAL UNIFIED PROCESS (RUP)

FIGURA1.
PROCESO UNIFICADO RATIONAL



[1]Roberth G. Figueroa, UTPL, Loja, rgfigueroa@utpl.edu.ec
[2] Camilo J. Solís, UTPL, Loja, cjsolis@utpl.edu.ec
[3] Armando Cabrera S,Docente-Investigador UTPL,Loja, aacabrera@utpl.edu.ec

RUP es un proceso formal: Provee un acercamiento disciplinado para asignar tareas y responsabilidades dentro de una organización de desarrollo. Su objetivo es asegurar la producción de software de alta calidad que satisfaga los requerimientos de los usuarios finales (respetando cronograma y presupuesto). Fue desarrollado por Rational Software, y está integrado con toda la suite Rational de herramientas. Puede ser adaptado y extendido para satisfacer las necesidades de la organización que lo adopte. (Customización). Es guiado por casos de uso y centrado en la arquitectura, y utiliza UML como lenguaje de notación.

Fases

Las cuatro fases del ciclo de vida son:

- Concepción
- Elaboración
- Construcción
- Transición

Ventajas

- Evaluación en cada fase que permite cambios de objetivos
- Funciona bien en proyectos de innovación.
- Es sencillo, ya que sigue los pasos intuitivos necesarios a la hora de desarrollar el software.
- Seguimiento detallado en cada una de las fases.

Desventajas

- La evaluación de riesgos es compleja
- Excesiva flexibilidad para algunos proyectos
- Estamos poniendo a nuestro cliente en una situación que puede ser muy incómoda para él.
- Nuestro cliente deberá ser capaz de describir y entender a un gran nivel de detalle para poder acordar un alcance del proyecto con él.

MICROSOFT SOLUTION FRAMEWORK (MSF)[4]

Descripción

MSF es un compendio de las mejores prácticas en cuanto a administración de proyectos se refiere. Más que una metodología rígida de administración de proyectos, MSF es una serie de modelos que puede adaptarse a cualquier proyecto de tecnología de información.

Todo proyecto es separado en cinco principales fases:

- Visión y Alcances.
- Planificación.
- Desarrollo.
- Estabilización.
- Implantación.

FIGURA 2.
MODELO DE EQUIPO DE MSF

Visión y Alcances:

La fase de visión y alcances trata uno de los requisitos más fundamentales para el éxito del proyecto, la unificación del equipo detrás de una visión común. El equipo debe tener una visión clara de lo que quisiera lograr para el cliente y ser capaz de indicarlo en términos que motivarán a todo el equipo y al cliente. Se definen los líderes y responsables del proyecto, adicionalmente se identifican las metas y objetivos a alcanzar; estas últimas se deben respetar durante la ejecución del proyecto en su totalidad, y se realiza la evaluación inicial de riesgos del proyecto.

Planificación:

Es en esta fase es cuando la mayor parte de la planeación para el proyecto es terminada. El equipo prepara las especificaciones funcionales, realiza el proceso de diseño de la solución, y prepara los planes de trabajo, estimaciones de costos y cronogramas de los diferentes entregables del proyecto.

Desarrollo:

Durante esta fase el equipo realice la mayor parte de la construcción de los componentes (tanto documentación como código), sin embargo, se puede realizar algún trabajo de desarrollo durante la etapa de estabilización en respuesta a los resultados de las pruebas. La infraestructura también es desarrollada durante esta fase.

Estabilización:

En esta fase se conducen pruebas sobre la solución, las pruebas de esta etapa enfatizan el uso y operación bajo condiciones realistas. El equipo se enfoca en priorizar y resolver errores y preparar la solución para el lanzamiento.

Implantación:

[4] Microsoft Solution Framework, (en línea), disponible en http://www.gpicr.com/msf.aspx

Durante esta fase el equipo implanta la tecnología base y los componentes relacionados, estabiliza la instalación, traspasa el proyecto al personal soporte y operaciones, y obtiene la aprobación final del cliente.

Modelo de roles

El modelo de equipos de MSF (MSF team model) fue desarrollado para compensar algunas de las desventajas impuestas por las estructuras jerárquicas de los equipos en los proyectos tradicionales.

Los equipos organizados bajo este modelo son pequeños y multidisciplinarios, en los cuales los miembros comparten responsabilidades y balancean las destrezas del equipo para mantenerse enfocados en el proyecto que están desarrollando. Comparten una visión común del proyecto y se enfocan en implementar la solución, con altos estándares de calidad y deseos de aprender.

El modelo de equipos de MSF tiene seis roles que corresponden a las metas principales de un proyecto y son responsables por las mismas. Cada rol puede estar compuestos por una o más personas, la estructura circular del modelo, con óvalos del mismo tamaño para todos los roles, muestra que no es un modelo jerárquico y que cada todos los roles son igualmente importantes en su aporte al proyecto. Aunque los roles pueden tener diferentes niveles de actividad durante las diversas etapas del proyecto, ninguno puede ser omitido.

La comunicación se pone en el centro del círculo para mostrar que está integrada en la estructura y fluye en todas direcciones. El modelo apoya la comunicación efectiva y es esencial para el funcionamiento del mismo

Ejemplo de metodología MSF aplicada

Como ejemplo de una aplicación de metodología MSF a un proyecto, a continuación se describe el contenido de cada una de las fases y, en la medida de lo posible, un detalle de acciones concretas y estimación de carga de trabajo en términos de jornadas, número de personas implicadas y perfil de las mismas. El proyecto ejemplo se trata de una implantación de infraestructuras, en concreto, migración a Windows 2000 de una red de servidores.

Fase 1 - Estrategia y alcance

En esta fase deberían tener lugar los siguientes trabajos:

- Elaboración y aprobación del Documento de Alcance y Estrategia definitivo: debe ser un documento de consenso con la participación del mayor número de agentes implicados en el proyecto. En este documento quedarán definitivamente reflejadas las funcionalidades y servicios que, ineludiblemente, debe ofrecer la solución a implantar.
- Formación del Equipo de Trabajo y distribución de competencias y responsabilidades: generalmente se definen como áreas principales la de Diseño de Arquitectura, Pruebas de Laboratorio, Documentación, Logística y Coordinación.
- Elaboración del Plan de Trabajo: deben marcarse fechas y contenidos para esta fase y las siguientes. Los mecanismos y protocolos de intercambio de información y coordinación deben quedar suficientemente bien establecidos y consensuados.
- Elaboración de la matriz de Riesgos y Plan de Contingencia: los principales riesgos detectados deben tener un plan de mitigación y actuación y revisarse con periodicidad.

Para un proyecto de migración a Windows 2000 podría estimarse que los trabajos indicados pueden requerir en torno a 20 jornadas de trabajo y la intervención de un Consultor de Microsoft junto con el equipo de trabajo que formen El cliente y el Partner.

Fase 2 - Planificación y Prueba de Concepto

Deben alcanzarse los siguientes objetivos e hitos:

- Documento de Planificación y Diseño de Arquitectura: es el documento principal, donde se describen en detalle los aspectos funcionales y operativos de la nueva plataforma. La aprobación de este documento es el hito principal de esta fase, y supone la directriz última de todos los trabajos técnicos, que, a partir de ese momento, deben ser consistentes con esta Guía. Si en el curso de las fases sucesivas fuera necesario revisar estos contenidos, se deberá hacer por acuerdo y conocimiento de todo el equipo de trabajo y se llevará un registro de versiones que permita hacer un seguimiento adecuado de estas revisiones.
- Documento de Plan de Laboratorio - Prueba de Concepto: la descripción del contenido del laboratorio de prueba de concepto, los diversos escenarios a simular, los criterios de validez, el control de incidencias y las métricas de calidad son objetivos a cubrir en este documento. Es un documento dinámico, en el que se recoge la idea y la experiencia práctica al llevarla a cabo en entorno controlado y aislado. La etapa de prueba de laboratorio concluye cuando la maqueta ofrece todos los servicios y funciones descritos en el Documento de Alcance y Estrategia, y su grado de estabilidad y rendimiento es considerado como "suficiente".

Habitualmente, en las propuestas de preventa no se pueden indicar con precisión parámetros como el esfuerzo técnico, tiempo o coste definitivo que puede suponer esta fase. De otras experiencias anteriores se puede obtener una relación de trabajos sólo a título orientativo, y que debe ser revisado y acordado por todas las partes:

El cómputo de jornadas para la relación de actividades descritas (que como se observa sólo recogen las relativas a la Planificación y Diseño, y deja aparte las necesarias para elaborar el plan de Migración), ofrecería este resultado:
Jornadas totales: 80 (aprox. 4 meses)
Jornadas / técnico del PARTNER: 150 jornadas (2 personas)
Jornadas / técnico del CLIENTE: 110 jornadas (Max. 2 personas)

Fase 3 – Estabilización

La solución implantada en la maqueta se pasa a un entorno real de explotación, restringido en número de usuarios y en condiciones tales que se pueda llevar un control efectivo de la situación. Los hitos y objetivos fundamentales de esta fase son:

- **Selección del entorno de prueba piloto:** se acordará la composición y ubicación del conjunto de máquinas y usuarios que entrarán en la prueba. Esta selección se recomienda que se haga atendiendo a la mayor variedad posible de casos, de manera que puedan aflorar el máximo de incidentes potenciales en el menor tiempo posible. La dimensión de la muestra tiene también que calcularse, sin perder de vista que la prueba piloto no es el despliegue propiamente, sino una fase de observación en la que es absolutamente crítico establecer unos cauces efectivos de tratamiento de los errores.
- **Gestión de Incidencias:** aunque esta labor se habrá iniciado en la fase anterior, el éxito de la prueba piloto dependerá de que se forme un sistema de recogida de incidentes (helpdesk o similar), de atención al usuario (formación, consultas) y de resolución de problemas y documentación de los mismos (versionado de la plataforma).
- **Revisión de la documentación final de Arquitectura:** el documento de Planificación y Diseño de Arquitectura se puede ver alterado parcialmente como resultado de esta fase. El documento final, aprobado por consenso, supone el principal documento del Proyecto y la culminación de los trabajos de diseño, al menos en sus líneas principales. Este documento se considerará definitivo cuando la solución puesta en marcha se muestre estable y el número de incidencias graves (de intervención o de resolución) sea nulo y la cantidad de las consideradas leves quede por debajo de un límite establecido en las Métricas de Calidad.
- **Elaboración de la documentación de Formación y Operaciones:** con vistas al soporte post proyecto y los programas de formación a usuarios y administradores, en esta fase deben elaborarse las Guías de Usuario, de Administración, las "paso-a-paso", y otros cuyos contenidos deben acordarse previamente.
- **Elaboración del Plan de Despliegue:** se debe consensuar la fecha de finalización de la fase Piloto, y las condiciones de calidad que debe cumplir la solución final para iniciar el despliegue. En el Plan deben identificarse las fases, estrategia de implantación, fechas, tareas a realizar, procedimientos de validación y método de control de incidencias.
- **Elaboración del Plan de Formación:** con anterioridad al despliegue definitivo, debe haberse aprobado el Plan de Formación orientado a usuarios finales y administradores, y debe hacerse compatible con los ritmos acordados en el Plan de Despliegue.

El tiempo necesario para abordar esta fase es variable y depende en parte de factores ajenos a la complejidad de la propia solución, como es la adecuada selección del entorno de prueba y el momento del año en que tenga lugar (evitando que coincida con periodos de vacaciones o puntas de trabajo críticas como Fin de Año). En proyectos de similar envergadura se ha llegado al momento "Error Free Versión" en 30 jornadas de trabajo (aproximadamente mes y medio) con una muestra de 50 usuarios.

Fase 4 – Despliegue

Se llevarán a cabo en esta fase los planes diseñados en la anterior, principalmente el de despliegue y el de formación. Los principales trabajos e hitos a conseguir son, en este caso, además de los obvios (implantación de la plataforma, puesta en servicio de todas las funciones, formación a los usuarios y administradores), los siguientes:

- Continuación con las labores de recepción de incidencias, clasificación, tratamiento, resolución y distribución de faxes o intervención on-site.
- Registro de mejoras y sugerencias, funcionalidades no cubiertas y novedades a incorporar en sucesivas versiones de la plataforma, incluyendo mejoras aportadas por los fabricantes de software (nuevas versiones o Service Packs, por ejemplo)
- Revisión de las Guías y manuales de usuario, rectificación de errores y obtención de los documentos de formación definitivos.
- Entrega de los documentos definitivos acordados como "deliverables" en la fase de Vision Scope.
- Revisión (si procede) de la matriz de riesgos, las métricas de calidad y establecimiento de los estándares de calidad y SLA definitivos.
- Finalmente, entrega del Proyecto y cierre del mismo, con o sin apertura de nuevo proyecto en base a la información y experiencia obtenidas.

La duración fase de despliegue, puesto que debe planificarse, no puede establecerse a priori. Depende de numerosos factores externos al propio proyecto (incluyendo factores de oportunidad política o de negocio) que pueden retardar o acelerar la conclusión.

La experiencia demuestra que no hay una relación directa entre número de máquinas y tiempo necesario para el despliegue. Los factores más relevantes en el cálculo suelen ser la dispersión o concentración geográfica, la complejidad del proceso de migración, el grado de automatización alcanzado, la experiencia y nivel de los técnicos que realizan la operación y condicionantes de calendario, a menudo con restricciones no técnicas, sino de otros tipos (las fechas-objetivo suelen marcarse por criterios de oportunidad de negocio).

METODOLOGÍAS ÁGILES.

Luego de varias opiniones tanto a favor como en contra de las metodologías tradicionales se genera un nuevo enfoque denominado, métodos ágiles, que nace como respuesta a los problemas detallados anteriormente y se basa en dos aspectos puntuales, el retrasar las decisiones y la planificación adaptativa; permitiendo potencia aún más el desarrollo de software a gran escala.

Como resultado de esta nueva teoría se crea un *Manifiesto Ágil*[5] cuyas principales ideas son:

- Los individuos y las interacciones entre ellos son más importantes que las herramientas y los procesos empleados.
- Es más importante crear un producto software que funcione que escribir documentación exhaustiva.
- La colaboración con el cliente debe prevalecer sobre la negociación de contratos.
- La capacidad de respuesta ante un cambio es más importante que el seguimiento estricto de un plan.

Entre los principales métodos ágiles tenemos el XP (eXtreme Programming), Scrum, Iconix, Cristal Methods, AUP entre otras.

Estas metodologías ponen de relevancia que la capacidad de respuesta a un cambio es más importante que el seguimiento estricto de un plan. Nos lo proponen porque para muchos clientes esta flexibilidad será una ventaja competitiva y porque estar preparados para el cambio significar reducir su coste.

Retrasar las decisiones y Planificación Adaptativa

Es el eje en cual gira la metodología ágil, el retrasar las decisiones tan como sea posible de manera responsable será ventajoso tanto para el cliente como para la empresa, lo cual permite siempre mantener una satisfacción en el cliente y por ende el éxito del producto, las principales ventajas de retrasar las decisiones son:

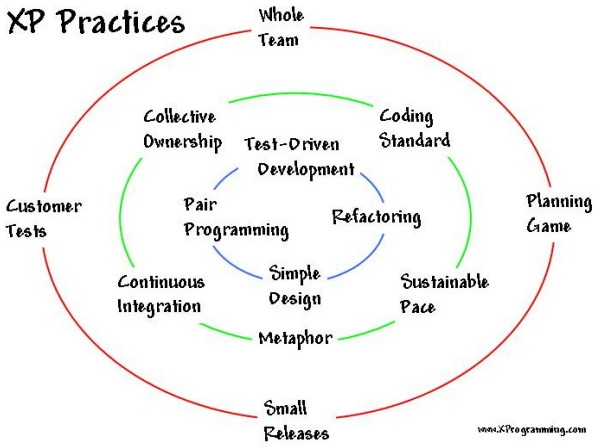
- ✓ Reduce el número de decisiones de alta inversión que se toman.
- ✓ Reduce el número de cambios necesario en el proyecto.
- ✓ Reduce el coste del cambio

La planificación adaptativa permite estar preparados para el cambio ya que lo hemos introducido en nuestro proceso de desarrollo, además hacer una planificación adaptativa consiste en tomar decisiones a lo largo del proyecto, estaremos transformando el proyecto en un conjunto de proyectos pequeños.

Esta planificación a corto plazo nos permitirá tener software disponible para nuestros clientes y además ir aprendiendo del feedback para hacer nuestra planificación más sensible, sea ante inconvenientes que aceleren o retrasen nuestro producto. A continuación se detalla el XP que es el más aceptado dentro del desarrollo de SW

EXTREME PROGRAMMING (XP)

FIGURA 3[6].
MODELO DE EXTREME PROGRAMMING



Es la más destacada de los procesos ágiles de desarrollo de software formulada por Kent Beck. La programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad.

Los defensores de XP consideran que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos. Creen que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos.

[5] Pires, Donald, "Manifiesto Ágil", UCLA, (en línea), disponible en http://www.manifiestoagile.com

[6] Extrem Porgramming(XP). Disponible en www.XProgramming.com

Las características fundamentales del método son:

- **Desarrollo iterativo e incremental:** pequeñas mejoras, unas tras otras.
- **Pruebas unitarias continuas**, frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión. Se aconseja escribir el código de la prueba antes de la codificación.
- **Programación por parejas**: se recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto. Se supone que la mayor calidad del código escrito de esta manera -el código es revisado y discutido mientras se escribe- es más importante que la posible pérdida de productividad inmediata.
- **Frecuente interacción** del equipo de programación con el cliente o usuario. Se recomienda que un representante del cliente trabaje junto al equipo de desarrollo.
- **Corrección** de todos los errores antes de añadir nueva funcionalidad. Hacer entregas frecuentes.
- **Refactorización** del código, es decir, reescribir ciertas partes del código para aumentar su legibilidad y mantenibilidad pero sin modificar su comportamiento. Las pruebas han de garantizar que en la refactorización no se ha introducido ningún fallo.
- **Propiedad del código compartida**: en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve el que todo el personal pueda corregir y extender cualquier parte del proyecto. Las frecuentes pruebas de regresión garantizan que los posibles errores serán detectados.
- **Simplicidad en el código**: es la mejor manera de que las cosas funcionen. Cuando todo funcione se podrá añadir funcionalidad si es necesario. La programación extrema apuesta que en más sencillo hacer algo simple y tener un poco de trabajo extra para cambiarlo si se requiere, que realizar algo complicado y quizás nunca utilizarlo.

La simplicidad y la comunicación son extraordinariamente complementarias. Con más comunicación resulta más fácil identificar qué se debe y qué no se debe hacer. Mientras más simple es el sistema, menos tendrá que comunicar sobre este, lo que lleva a una comunicación más completa, especialmente si se puede reducir el equipo de programadores.

Ventajas

- Apropiado para entornos volátiles
- Estar preparados para el cambio, significa reducir su coste.
- Planificación más transparente para nuestros clientes, conocen las fechas de entrega de funcionalidades. Vital para su negocio
- Permitirá definir en cada iteración cuales son los objetivos de la siguiente
- Permite tener realimentación de los usuarios muy útil.
- La presión esta a lo largo de todo el proyecto y no en una entrega final

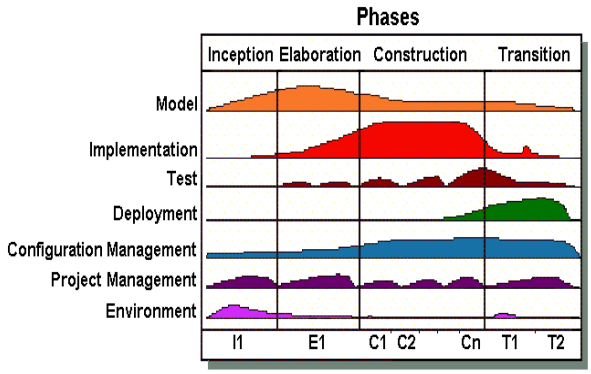
Desventajas

- Delimitar el alcance del proyecto con nuestro cliente

Para mitigar esta desventaja se plantea definir un alcance a alto nivel basado en la experiencia.

AUP (AGIL UNIFIED PROCESS)

FIGURA 4.
ESQUEMA DE TRABAJO AUP

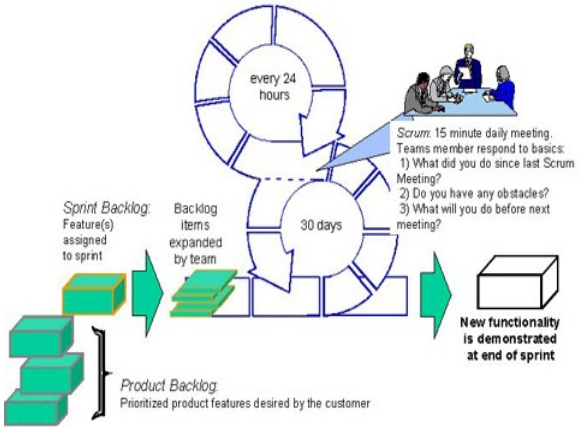


El AUP es un acercamiento aerodinámico a desarrollo del software basado en el Proceso Unificado Rational de IBM (RUP), basado en disciplinas y entregables incrementales con el tiempo. El ciclo de vida en proyectos grandes es serial mientras que en los pequeños es iterativo. Las disciplinas de Aup son:

- Modelado
- Implementación
- Prueba
- Despliegue
- Administración de la configuración
- Administración o gerencia del Proyecto
- Entorno

SCRUM

FIGURA 5.
ESQUEMA DE TRABAJO SCRUM



Scrum es un proceso ágil y liviano que sirve para administrar y controlar el desarrollo de software. El desarrollo se realiza en forma iterativa e incremental (una iteración es un ciclo corto de construcción repetitivo). Cada ciclo o iteración termina con una pieza de software ejecutable que incorpora nueva funcionalidad. Las iteraciones en general tienen una duración entre 2 y 4 semanas. Scrum se utiliza como marco para otras prácticas de ingeniería de software como RUP o Extreme Programming.

Scrum se focaliza en priorizar el trabajo en función del valor que tenga para el negocio, maximizando la utilidad de lo que se construye y el retorno de inversión. Está diseñado especialmente para adaptarse a los cambios en los requerimientos, por ejemplo en un mercado de alta competitividad. Los requerimientos y las prioridades se revisan y ajustan durante el proyecto en intervalos muy cortos y regulares. De esta forma se puede adaptar en tiempo real el producto que se está construyendo a las necesidades del cliente. Se busca entregar software que realmente resuelva las necesidades, aumentando la satisfacción del cliente.

En Scrum, el equipo se focaliza en una única cosa: construir software de calidad. Por el otro lado, la gestión de un proyecto Scrum se focaliza en definir cuales son las características que debe tener el producto a construir (qué construir, qué no y en qué orden) y en remover cualquier obstáculo que pudiera entorpecer la tarea del equipo de desarrollo. Se busca que los equipos sean lo más efectivos y productivos posible.

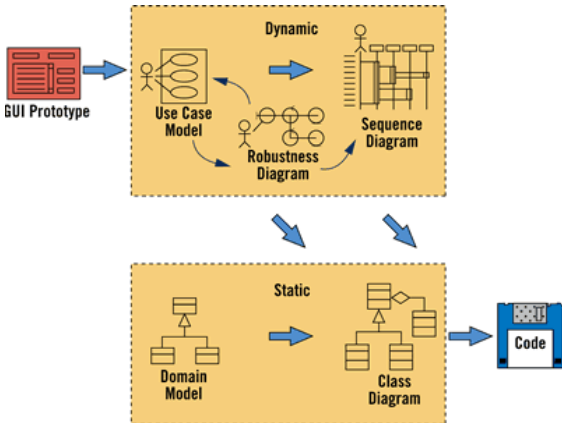
Scrum tiene un conjunto de reglas muy pequeño y muy simple y está basado en los principios de inspección continua, adaptación, auto-gestión e innovación. El cliente se entusiasma y se compromete con el proyecto dado que ve crecer el producto iteración a iteración y encuentra las herramientas para alinear el desarrollo con los objetivos de negocio de su empresa.

Por otro lado, los desarrolladores encuentran un ámbito propicio para desarrollar sus capacidades profesionales y esto resulta en un incremento en la motivación de los integrantes del equipo.

ICONIX

El proceso de ICONIX maneja casos de uso, como el RUP, pero le falta mucho para llegar al nivel del RUP. También es relativamente pequeño y firme, como XP, pero no desecha el análisis y diseño que hace XP. Este proceso también hace uso aerodinámico del UML mientras guarda un enfoque afilado en el seguimiento de requisitos.

FIGURA 6.
ESQUEMA DE TRABAJO ICONIX



Y, el proceso se queda igual a la visión original de Jacobson del manejo de casos de uso, esto produce un resultado concreto, específico y casos de uso fácilmente entendible, que un equipo de un proyecto puede usar para conducir el esfuerzo hacia un desarrollo real. La Figura 6 muestra el cuadro del proceso. El diagrama retrata la esencia del enfoque aerodinámico al desarrollo del software, que incluye un juego mínimo de diagramas de UML y algunas valiosas técnicas que se toman de los casos del uso para codificar rápida y eficazmente. El enfoque es flexible y abierto; siempre se puede seleccionar de los otros aspectos del UML para complementar los materiales básicos.

1. Diferencias:

TABLA 1.
DIFERENCIAS ENTRE METODOLOGÍA TRADICIONALES Y ÁGILES

Metodologías Tradicionales	Metodologías Agiles
Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo	Basadas en heurísticas provenientes de prácticas de producción de código
Cierta resistencia a los cambios	Especialmente preparados para cambios durante el proyecto
Impuestas externamente	Impuestas internamente (por el equipo)
Proceso mucho más controlado, con numerosas políticas/normas	Proceso menos controlado, con pocos principios.

El cliente interactúa con el equipo de desarrollo mediante reuniones	El cliente es parte del equipo de desarrollo
Más artefactos	Pocos artefactos
Más roles	Pocos roles
Grupos grandes y posiblemente distribuidos	Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio
La arquitectura del software es esencial y se expresa mediante modelos	Menos énfasis en la arquitectura del software
Existe un contrato prefijado	No existe contrato tradicional o al menos es bastante flexible

Ofrecemos una comparativa entre cada uno de las etapas más comunes del desarrollo de SW y los enfoques de metodologías revisados.

TABLA 2.
DIFERENCIAS POR ETAPAS Y ENFOQUE METODOLOGICO

MODELOS RIGUROSOS	ETAPA	MODELOS AGILES
Planificación predictiva y "aislada"	Análisis de requerimientos	Planificación adpatativa:Entregas frecuentes + colaboración del cliente
	Planificación	
Diseño flexible y Extensible + modelos + Documentación exhaustiva	Diseño	Diseño Simple: Documentación Mínima + Focalizado en la comunicación
Desarrollo individual con Roles y responsabilidades estrictas	Codificación	Transferencia de conocimiento: Programación en pares + conocimiento colectivo
Actividades de control]: Orientado a los hitos + Gestión miniproyectos	Pruebas Puesta en Producción	Liderazgo-Colaboración: empoderamiento +auto-organización

Además presentamos un cuadro de comparación por cada aspecto analizado y lo ponemos a consideración:

- Por las características del Proyecto

Modelo de Proceso	Tamaño del Proceso	Tamaño del Equipo	Complejidad del Problema
RUP	Medio / Extenso	Medio / Extenso	Medio / Alto
ICONIX	Pequeño / Medio	Pequeño / Medio	Pequeño / Medio
XP	**Pequeño / Medio**	**Pequeño**	Medio / Alto
SCRUM	**Pequeño / Medio**	**Pequeño**	Medio / Alto

En este cuadro se presenta una comparativa de las modelos de proceso en cuanto a las características del proyecto, analizamos el tamaño del proceso, del equipo y la complejidad del problema para cada uno de los modelos. Podemos resaltar que: con un pequeño equipo de desarrollo se puede realizar grandes proyectos, de alta complejidad; es el caso de XP y SCRUM.

- Por la curva de Aprendizaje

Modelo de Proceso	Curva de aprendizaje	Herramienta de integración	Soporte Externo
RUP	**Lenta**	**Alto Soporte**	**Alto Soporte**
ICONIX	**Rápida**	Algún Soporte Disponible	Algún Soporte Disponible
XP	Rápida	No mencionado	Algún Soporte Disponible
SCRUM	Rápida	No mencionado	Algún Soporte Disponible

Con respecto a la curva de aprendizaje, vemos que los modelos ágiles, nos ofrecen una mayor ventaja pero con ciertas limitaciones, ya que aun no hay sido explotadas a gran escala como lo es RUP que posee alto soporte y herramientas integrales que nos guían a través del mismo, facilitando aplicar con mayor efectividad esta metodología, permitiendo aprovecharla al máximo

CONCLUSIONES

- El retrasar las decisiones en un proyecto de software permite potenciar el valor del producto tanto para el cliente como al equipo o empresa que desarrolla
- Para que un grupo de desarrollo adopte una metodología ágil debe poseer experiencia trabajando con metodologías tradicionales, ya que la experiencia es la que predomina en los mementos cruciales del proyecto, además debe tener la capacidad de ser equipos auto-gestionados, altamente motivados y con gran innovación

- Las metodologías ágiles permiten disminuir costos y brindar flexibilidad a los proyectos de software donde la incertidumbre está presente
- El uso de metodologías tradicionales es esencial al inicio en un equipo de desarrollo de software
- Las metodologías ágiles se deberían aplicar en proyectos donde exista mucha incertidumbre donde el entorno es volátil, donde los requisitos no se conocen con exactitud, mientras que las metodologías

tradicionales obligan al cliente a tomar las decisiones al inicio del proyecto.

REFLEXIÓN

- **¿Alguna recomendación para los lectores de este Artículo?**

gran giro a la industria del software. Por último, que entiendan la importancia de la gente. Con este fin voy a hacer una última analogía: el dueño de una fábrica de coches sale a las 6 de la tarde, y ahí tiene su fábrica, con su valor intacto; puede venderla y recuperar su inversión. En cambio, el dueño de una fábrica de software, a las 8 de la noche que sus empleados ya se fueron a su casa, está descapitalizado. Lo único que tiene son escritorios y unas máquinas depreciadas.

Como industria, debemos reconocer que estamos hechos de personas, y que éstas son nuestro principal activo. Así que debemos tenerlas bien "aceitadas" (a través de capacitación y motivación) para obtener su máximo rendimiento.

BIBLIOGRAFÍA

[1] Metodologías Ágiles: La ventaja competitiva de estar preparado para tomar decisiones lo más tarde posible y cambiarlas en cualquier momento. (En línea), Disponible en: **www.spinec.org/wp-content/metodologiasagiles_01.pdf**

[2] Metodologías ágiles (En línea) ,Disponible en: **http://www.agile-spain.com**

[3] ICONIX (En línea), Disponible en: **www.spinec.org/wp-content/ICONIX.pdf**

[4] Extreme Programming Refactored: The Case Against XP, MATT Stephens and DOUG Rosenberg, Disponible en Formato chm

[5] Introducción a Iconix (En línea), Disponible en: **http://www.informit.com/articles/article.asp?p=167902&rl=1**

Primero, que conozcan y apliquen las técnicas de ingeniería de software. Segundo, que se incorporen de lleno métodos de calidad en sus procesos y que la forma más eficiente y efectiva de hacer las cosas, es hacerlas bien a la primera vez y con ello daremos un