DESARROLLO ADAPTATIVO DE SOFTWARE

Luis Acevedo

Postgrado en Ciencias de la Computación

Facultad de Ciencias de la Universidad Central de Venezuela

Caracas, Venezuela

laar@protonmail.com

Resumen—El objetivo de este artículo es describir la metodología Adaptative Software Development (Desarrollo Adaptativo de Software). Esta metodología está pensada para la construcción de sistemas de software grandes y complejos mediante un enfoque ágil.

I. INTRODUCCIÓN

La mayoría de los productos de software se desarrollan para algún cliente en particular o para un mercado en general. En los años 50, no existían metodologías de desarrollo, éste estaba a cargo de los propios programadores, de ahí la importancia de contar con analistas y diseñadores que permitieran un análisis adecuado de las necesidades que se deberían implementar; aun así, los resultados eran impredecibles, no se sabía la fecha exacta cuando concluiría un proyecto de software, ni había forma de controlar las actividades que se estaban desarrollando; tampoco se contaba con documentación estandarizada. Debido a ésto, nacen muchos modelos y metodologías para el desarrollo de software.

Existen las llamadas metodologías pesadas, como Rational Unified Process (RUP) o Microsoft Solutions Framework (MSF), entre otras; y las ágiles, como lo son Scrum, Extreme Programming, Adaptative Software Development y muchas más, diseñadas para servir de guía durante el ciclo de vida de desarrollo de un sistema de software. Cada una posee sus ventajas y desventajas frente a distintos escenarios.

Los procesos de desarrollo de software basados en una completa especificación de los requerimientos de diseño, construcción y pruebas del sistema, no se ajustan al desarrollo rápido de aplicaciones. Cuando los requerimientos cambian o se descubren problemas con ellos, el diseño o implementación del sistema se tiene que realizar de nuevo y ser probado. Como consecuencia, normalmente, se

prolonga en el tiempo un proceso en cascada convencional, y el software definitivo se entrega mucho tiempo después con el que inicialmente se pactó.

Actualmente, el desarrollo y las entregas rápidas son los requerimientos más críticos de los sistemas. Estas entregas rápidas de software necesitan metodologías ligeras o ágiles. Dichas metodologías combinan una filosofía y un conjunto de directrices de desarrollo que buscan la satisfacción del cliente y la entrega temprana de software incremental; todo esto, mediante la flexibilidad en los procesos, la conformación de equipos pequeños de trabajo con alta motivación, el uso de métodos informales y una simplicidad general en las fases del desarrollo. Los procesos de desarrollo ágiles de software están diseñados para producir software útil e incremental, de forma rápida. Estos procesos son interactivos, es decir, se entrelazan la especificación, el diseño, el desarrollo y las pruebas.

En el desarrollo de software, el término "ágil" se concibe como "la capacidad para responder a los cambios", bien sea cambios de requisitos, de tecnología, de presupuesto, de cronograma, de personal, entre otros. Es por ello que las metodologías ágiles compartes ciertos aspectos:

- Formar equipos multidisciplinarios, multi-funcionales, con poder y auto-organización.
- Fomentar la responsabilidad compartida y la rendición de cuentas.
- Estimular la colaboración con el cliente.
- Cultivar la comunicación de todo el equipo para evitar retrasos y tiempos de espera.
- Priorizar las entregas frecuentes y continuas, para asegurar una rápida retroalimentación, permitiendo al equipo alinearse a los requerimientos.
- Incentivar la colaboración, ésto facilita la combinación de diferentes perspectivas oportunas en la implementación, la corrección de defectos y la adaptación a los cambios.

De acuerdo con lo anteriormente planteado, los métodos ágiles están basados en una reducción del ciclo de desarrollo, el cual se repite con frecuencia; cada ciclo se llama iteración, pudiendo el nuevo software ser lanzado al final de una sola iteración o después de varias. Los métodos ágiles minimizan el riesgo de fracaso en los proyectos que requieren de entregas rápidas. La clave para el desarrollo iterativo es producir, frecuentemente, versiones de trabajo del sistema final, que tengan un subconjunto de las características requeridas, integradas y probadas.

Los metodologías ágiles adoptan ciertas técnicas con la finalidad de mejorar la calidad, tanto del producto como del proceso en general, a lo largo del ciclo de desarrollo. Algunas de estas técnicas son: la programación en pareja, la realización de pruebas de aceptación, desarrollo orientado a pruebas (TDD), y la constante integración para ayudar a mantener la calidad del código; siendo importantes las reuniones de revisión al finalizar cada iteración.

En el presente trabajo, se hará una concepción de la metodología Adaptative Software Development y se detallarán sus ciclos o fases. En la segunda parte, se hablará más profundamente sobre esta metodología, en qué consiste y su origen. En la tercera parte, se explicará, detalladamente, su ciclo de vida**.** En la cuarta parte, se dará una conclusión basada en lo anteriormente expuesto.

Esta investigación se realizó mediante la búsqueda de artículos relacionados al tema, extrayendo lo esencial de cada uno. Es de mencionar que existen modificaciones durante la redacción, por motivo de traducción, mas, no se pierde su idea y esencia original.

II. DESARROLLO ADAPTATIVO DE SOFTWARE (DAS)

La metodología Adaptative software Development (ASD), en español Desarrollo Adaptativo de Software (DAS), fue propuesta por Jim Highsmith en el año 1998, como una técnica para construir software y sistemas complejos mediante un enfoque ágil o ligero; proviene del proceso de desarrollo rápido de aplicaciones propuesto por el mismo Jim Highsmith junto con Sam Bayer. Esta metodología permite la adaptación del desarrollo de software, por lo que su fundamento es adaptarse a los cambios y no ir en contra de ellos, todo esto, enmarcado en el principio de la continua adaptación del proceso de desarrollo al trabajo real. Su funcionamiento es cíclico al igual que otras metodologías, pero diverge en que no tiene un ciclo de planificación-diseño-construcción como la mayoría de ellas, en su lugar, tiene el ciclo especular-colaborar-aprender, debido a que se reconoce que en cada iteración se producirán cambios y errores. Un ciclo de vida del DAS debe enfocarse en procesos iterativos, planeados de acuerdo al tiempo y guiados por los riesgos, teniendo siempre presente la tolerancia al cambio. Los apoyos filosóficos del DAS se enfocan en la colaboración humana y la organización propia del equipo.

El DAS proporciona un marco para el desarrollo iterativo de sistemas grandes y complejos, fomentando el desarrollo incremental mediante el uso de prototipos. Esta metodología resalta que las aproximaciones en cascada sólo funcionan en entornos bien conocidos. DAS se define como una metodología tolerante a los cambios que ocurren frecuentemente en el desarrollo de software.

Esta metodología de desarrollo de software surgió debido a la necesidad de adaptarse a los cambios, en vez de revertirlos o prevenirlos, guiándose por el principio de la adaptación continua. En otras palabras, adaptarse al cambio en lugar de luchar contra él.

A diferencia de la mayoría de las metodologías de desarrollo de software, las cuales utilizan un ciclo de vida orientado a diseñar-construir, DAS ofrece un ciclo de vida en el cual cada ciclo se puede iterar; este ciclo de desarrollo dinámico está dedicado completamente al aprendizaje y a una intensa colaboración entre los desarrolladores y los clientes, esto debido al constante cambio en el ambiente de los negocios.

El Desarrollo Adaptativo de Software se centra en la colaboración y el aprendizaje como técnica para construir sistemas complejos; evolucionada de las mejores prácticas del Desarrollo Rápido de Aplicaciones (RAD) y de los Ciclos de vida evolutivos, ampliándose para incluir enfoques adaptativos para la gestión, sustituyendo la planificación por la especulación.

En palabras de Highsmith:

> "El Desarrollo Adaptativo de Software es cíclico como el modelo evolutivo, con los nombres de especular, colaborar y aprender, reflejando el carácter impredecible de sistemas cada vez más complejos; el Desarrollo Adaptativo va más allá de su herencia evolutiva: primero, reemplaza explícitamente el determinismo con el surgimiento; segundo, va más allá de un cambio en el Ciclo de Vida a un cambio más profundo en el estilo de gestión".

El Desarrollo Adaptativo de Software reemplaza el ciclo tradicional de cascada con una serie repetitiva de ciclos de especular, colaborar y aprender. Estos ciclos dinámicos proporcionan un aprendizaje continuo y adaptación a los posibles estados emergentes del proyecto. En principio, se centra en la rápida creación y evolución de sistemas de software; no existe un período en el que el software está terminado, sólo hay períodos estables entre nuevas versiones, a diferencia de su predecesora (el Método de Desarrollo Rápido de Aplicaciones) que permite un momento en el que el proyecto está terminado, mientras que el desarrollo adaptativo de software no.

DAS está enfocada y dirigida para el desarrollo de productos de software complejos y sistemas grandes, mediante el desarrollo iterativo con la creación de prototipos.

III. Ciclo de vida del DAS

El Desarrollo Adaptativo de Software es cíclico, al igual que el modelo Evolutivo, con los nombres de las fases reflejando la impredecibilidad de los sistemas complejos: especular, colaborar, aprender. Estas tres fases reflejan su naturaleza dinámica. El Desarrollo Adaptativo de Software reemplaza explícitamente lo determinístico con lo emergente, y va más allá de un simple cambio en el ciclo de vida, hace un cambio profundo en el estilo de gestión.

El ciclo de vida del desarrollo adaptativo de software se centra en los resultados, no en las tareas, y los resultados se identifican como características de la aplicación.

Etapas del DAS:

1. Especular

En esta etapa, se realiza una planificación tentativa del proyecto en función de las entregas (incrementos) que se irán realizando, estableciendo los objetivos y las metas generales, tomando en cuenta sus limitaciones y riesgos. Es necesario fijar un rumbo determinado a seguir, estimando los tiempos para el desarrollo de las actividades, por lo que se enumera el número de iteraciones (tentativas), y se resaltan los objetivos prioritarios para la iteración actual.

Es especulativo debido los distintos escenarios que se plantean durante esta fase y la manera de abordarlos, mientras que al mismo tiempo, se va definiendo el proyecto. La planificación del ciclo adaptativo utiliza la información inicial del proyecto (fechas límites y descripciones del usuario) y los requerimientos básicos para definir el conjunto de ciclos de lanzamientos (o incrementos del software) que serán requeridos para el proyecto.

La completa planificación resulta muy determinista y rígida para esta metodología, restringiendo la capacidad de tomar rutas alternas y hasta innovadoras. Es por ello que el término “plan” se reemplaza por especular, aunque no deja de ser un tipo de planificación flexible, que reconoce la realidad y la incertidumbre en los problemas complejos.

Esta fase tiene cinco prácticas que pueden ser ejecutadas repetidamente durante la etapa de iniciación y planificación:

- Inicio del proyecto.
- Establecimiento de una caja de tiempo para todo el proyecto.
- Definición del número de iteraciones y la casilla de tiempo de cada una de ellas.

- Desarrollo de un tema u objetivo para cada una de las iteraciones.
- Asignación de características a cada iteración.

2. Colaborar

La colaboración se refiere al equilibrio en la asignación del trabajo, tomando en cuenta diversos factores y sus posibles inconvenientes, como la tecnología, los requisitos, las partes interesadas, entre otras. Los ciclos de aprendizaje están basados en pequeñas interacciones con el diseño, la construcción y las pruebas, durante las cuales el conocimiento se acumula, cometiendo pequeños errores basados en suposiciones falsas y corrigiendo esos errores, lo que conduce a una mayor experiencia y un mejor dominio del problema.

Analizado desde otra perspectiva, las aplicaciones complejas no se construyen, sino que evolucionan, y requieren que se recopile, analice y aplique un gran volumen de información al problema, ya que los entornos cambiantes tienen altos índices de flujo de información, dando como resultado diversos conocimientos que sólo pueden ser manejados mediante la colaboración de un equipo.

Colaborar requiere de la capacidad de trabajar conjuntamente para producir resultados, compartir conocimientos o tomar decisiones. En esta segunda etapa, se construye la funcionalidad definida durante la especulación. Durante cada iteración, el equipo colabora intensamente para liberar la funcionalidad planificada, aquí se abarca la posibilidad de explorar nuevas alternativas. Se hace énfasis en el aprendizaje colectivo del grupo de desarrollo. Se busca que el equipo no sólo se comunique o se encuentre completamente integrado, se desea que exista confianza, donde se puedan realizar críticas constructivas, y desarrollar actitudes que contribuyan al trabajo que se encuentran realizando. En esta fase, es imperante balancear el trabajo.

La comunicación, el trabajo en equipo y la creatividad individual son parte de la colaboración efectiva. Los miembros de los equipos trabajan simultáneamente para entregar los componentes del sistema. Durante esta fase, el foco está en el desarrollo y la integración de los componentes del software. La importancia radica en el aprendizaje y en el progreso a través de un ciclo completo. De hecho, Highsmith argumenta que los desarrolladores de software, a menudo, sobre-estiman la comprensión de los proyectos, y que el aprendizaje les podrá ayudar a mejorar su grado de entendimiento real. Los equipos del DAS aprenden de tres maneras:

1. Grupos enfocados. El cliente o los usuarios finales proporcionan

retroalimentación sobre los incrementos de software que son entregados. Esto indica, de forma directa, la satisfacción o la insatisfacción de las necesidades del negocio.

2. Revisiones técnicas formales. Los componentes del software desarrollado son revisados por los miembros del equipo, mientras su calidad y aprendizaje son mejorados.
3. Post mortem. El equipo de DAS se vuelve introspectivo al vigilar su propio desempeño, con el propósito de aprender acerca de su enfoque y, después, mejorarlo.

La colaboración es una actividad multidisciplinaria y compartida que engloba al equipo de desarrollo, a los clientes y a los gerentes.

3. Aprender

Por último, tenemos la etapa de aprendizaje. Consiste en la revisión de calidad que se realiza al final de cada ciclo: calidad del resultado desde la perspectiva del cliente, calidad del resultado desde la perspectiva técnica, calidad del funcionamiento del equipo de desarrollo y las prácticas que éste utiliza, al igual que se actualiza el estatus del proyecto. A su vez, se evalúa el conocimiento constantemente, realizando retroalimentaciones y reuniones de grupo al final de cada ciclo iterativo, ya que ésto ayuda a soportar y solucionar de una mejor manera el constante cambio que puede tener el proyecto y su adaptación. Permite mejorar el entendimiento real sobre la tecnología y los procesos utilizados en el proyecto.

Esta fase consiste en la revisión de calidad. Durante la fase de aprendizaje, se libera a los usuarios la última versión del software, se generan los informes, y se reinicia el ciclo. Aquí se involucran a todas las partes interesadas junto a los miembros del equipo del proyecto. Las posibles soluciones propuestas deben permitir a los usuarios adaptar sus métodos al mismo ritmo que el desarrollo del software, dando un cambio suave y bien entendido en el sistema. Las iteraciones deben ser cortas, para que el equipo pueda aprender de los errores pequeños en lugar de los grandes.

Al final de cada iteración de desarrollo, hay cuatro categorías generales de tópicos a tomar en cuenta y de los cuales se puede aprender:

1. Calidad de los resultados desde la perspectiva del cliente:

En los proyectos DAS, la prioridad es obtener retroalimentación de los clientes. La práctica recomendada para ello es crear un grupo de enfoque al cliente. Estas sesiones están diseñadas para registrar las solicitudes de cambio de los clientes y para revisar la aplicación de software en sí.

2. Calidad de los resultados desde el punto de vista técnico:

Se le debe dar importancia a la revisión de los aspectos técnicos a implementar, así como también, la revisión de las tecnologías a utilizar; esto puede realizarse semanalmente o al final de una iteración.

3. Calidad del funcionamiento del equipo de desarrollo y pruebas:

Los equipos deben monitorizar su propio desempeño periódicamente y las retrospectivas los animan a auto evaluarse y a aprender acerca de su propio trabajo, facilitando la revisión periódica del rendimiento conjunto, haciendo énfasis en:

- Determinar lo que no está funcionando.
- Lo que el Equipo necesita hacer más.
- Lo que el Equipo necesita hacer menos.

4. Estado del proyecto:

La determinación del estado del proyecto es un enfoque basado en características. La revisión del estado del proyecto debe incluir:

- ¿Dónde está el proyecto?.
- ¿Dónde está el proyecto frente a los planes?.
- ¿Dónde debería estar el proyecto?.

Es menester destacar que las tres fases no son lineales y se superponen. La parte de aprender del ciclo de vida es vital para el éxito del proyecto. Tanto los desarrolladores como los clientes examinan sus suposiciones y utilizan los resultados de cada ciclo de desarrollo para establecer la dirección del siguiente.

IV. CONCLUSIONES

Cada metodología tiene ventajas y desventajas que dependen del entorno donde se utilicen, del equipo de trabajo para que la metodología apropiada logre las metas, los objetivos a alcanzar en un software a construir, así como la planificación y el tiempo; siendo estos factores importantes para la ejecución del proyecto.

Por otro lado, las actividades de cada ciclo de desarrollo deben estar justificadas con respecto a la misión general del proyecto durante la implementación de esta metodología, por lo que se puede decir que está “centrada en la misión”. Para muchos proyectos, la misión general que guía al equipo está bien articulada, aunque los requisitos pueden ser inciertos al principio del proyecto. Una misión proporciona límites en lugar de un destino fijo, esto señala la dirección y criterios para la toma de decisiones críticas. Sin una misión clara y una práctica constante de refinamiento de la misión, los ciclos de vida iterativos se convierten en ciclos de vida oscilantes, balanceándose hacia adelante y hacia atrás sin progreso en el desarrollo. También, la funcionalidad que se desarrolla durante una iteración está basada en las

prioridades del cliente, pudiendo estas prioridades, evolucionar a lo largo de varias iteraciones cuando los clientes proporcionan retroalimentación, generando resultados directos al cliente después de la implementación.

A la postre, el ciclo de vida de esta metodología resulta iterativo, centrándose en las entregas incrementales frecuentes para obtener retroalimentación, asimilando el aprendizaje resultante y estableciendo la dirección correcta para el desarrollo posterior. Y gracias a su tolerancia al cambio como valor agregado, lo incorpora y concibe como una ventaja competitiva y no como un problema; abordando y monitorizando los riesgos desde el momento en que se detectan, conduciendo las iteraciones por la identificación y evaluación de los riesgos críticos.

La metodología de Desarrollo Adaptativo de Software fomenta el concepto de experimentar y aprender. Siendo una opción recomendable para construir grandes sistemas en los que no se vislumbre su total capacidad. Debido a las interacciones entre los miembros del equipo se pueden encontrar errores tempranos, aprender de ellos y evitar el re-trabajo, identificando pequeños problemas antes de que se conviertan en grandes problemas.

REFERENCIAS

[1] “Adaptive S/W Development – Introduction”. https://www.tutorialspoint.com/adaptive_software_development/adaptive_software_development_quick_guide.htm. Web. Consultado el 20 de julio del 2018.

[2] “Desarrollo adaptativo de software”. https://es.wikipedia.org/wiki/Desarrollo_adaptativo_de_software. 10 de mayo del 2018. Web. Consultado el 20 de julio del 2018.

[3] Peter Norvig and David Cohn. “ADAPTIVE SOFTWARE”. https://www.norvig.com/adapaper-pcai.html. Web. Consultado el 20 de julio del 2018.

[4] “Looking At Adaptive Software Development Information Technology Essay”. https://www.ukessays.com/essays/information-technology/looking-at-adaptive-software-development-information-technology-essay.php. 23 de marzo del 2015 . Web. Consultado el 20 de julio del 2018.

[5] Victor M. Font Jr. “Adaptive Software Development”. https://ultimatesdlc.com/adaptive-software-development/. Web. Consultado el 20 de julio del 2018.

[6] “Adaptive Software Development - Lifecycle”. https://www.tutorialspoint.com/adaptive_software_development/adaptive_software_development_lifecycle.htm. Web. Consultado el 20 de julio del 2018.

[7] “Adaptive Software Development”. http://adaptivesoftwaredevelopment.wikidot.com/. Web. Consultado el 20 de julio del 2018.

[8] John Rojas Hernández. “D.A.S. (Desarrollo Adaptativo de Software).”. https://prezi.com/numbr-9bpnqn/das-desarrollo-adaptativo-de-software/. 21 de mayo del 2014. Web. Consultado el 20 de julio del 2018.

[9] “ASD (Adaptive Software Development)”. https://adaptivesoftwaredevelopment.blogspot.com/. 13 de junio del 2012. Web. Consultado el 20 de julio del 2018.

[10] Diego Andrés González Moreno, José Luis Perea Álvarez. “DESARROLLO ADAPTABLE DE SOFTWARE, UNA SOLUCIÓN ÁGIL PARA APLICACIONES E-COMMERCE”. https://www.javeriana.edu.co/biblos/tesis/ingenieria/Tesis192.pdf. 2005. Web. Consultado el 20 de julio del 2018.

[11] Ken Orr. “Chapter 23. Adaptive Software Development”. https://www.exa.unicen.edu.ar/catedras/agilem/cap23asd.pdf. Web. Consultado el 20 de julio del 2018.

[12] James A. Highsmith III. “Adaptive Software Development, A COLLABORATIVE APPROACH TO MANAGING COMPLEX SYSTEMS”. https://books.google.co.ve/books?hl=es&lr=&id=CVcUAAAAQBAJ&oi=fnd&pg=PR7&dq=adaptive+software+development&ots=5szD6yJCyN&sig=d89T6eimcOM4dllonXdKnfj9QqU#v=onepage&q=adaptive%20software%20development&f=false. Web. Consultado el 21 de julio del 2018.