Propuesta Metodológica para Desarrollo Ágil de Software

Autores

JIMÉNEZ REY, M. Elizabeth (ejimenez@mara.fi.uba.ar)

GROSSI, María Delia (mdgrossi@mara.fi.uba.ar)

SERVETTO, Arturo Carlos (aserve@mara.fi.uba.ar)

PERICHINSKY, Gregorio (gperi@mara.fi.uba.ar)

Paseo Colón N° 850, 4° Piso, Tel. 4343-0891 Int. 142

Departamento de Computación, Facultad de Ingeniería, Universidad de Buenos Aires

Introducción

Se presenta el contexto en el que surgen las metodologías ágiles, sus valores, principios y comparación con las metodologías tradicionales.

Existen numerosas propuestas metodológicas para el desarrollo de software que inciden en distintas dimensiones del proceso de desarrollo.

Las propuestas más tradicionales se centran especialmente en el control del proceso, mediante una rigurosa definición de roles, de las actividades involucradas , de los artefactos que se deben producir y de las herramientas y notaciones que se usarán, incluyendo modelado y documentación detallada.

Este esquema para abordar el desarrollo de software ha demostrado ser efectivo y necesario en proyectos de gran tamaño (respecto a tiempo y recursos), donde por lo general, se exige un alto grado de ceremonia en el proceso. Sin embargo, este enfoque no resulta ser el más adecuado para muchos de los proyectos actuales, donde el entorno del sistema es muy cambiante y en donde se exige reducir drásticamente los tiempos de desarrollo, pero manteniendo una alta calidad.

En este escenario, las metodologías ágiles emergen como una posible respuesta para llenar ese vacío metodológico. Por estar especialmente orientadas para proyectos pequeños, las metodologías ágiles constituyen una solución a medida para ese entorno, aportando una elevada simplificación, que a pesar de ello, no renuncia a las prácticas esenciales para asegurar la calidad del producto.

En la comunidad de la ingeniería del software se está viviendo con intensidad un debate abierto entre los partidarios de las metodologías tradicionales (referidas peyorativamente como "metodologías pesadas") y aquellos que apoyan las ideas emanadas del "Manifiesto Ágil". [1]

El Manifiesto Ágil

En febrero de 2001, tras una reunión celebrada en Utah-EEUU, nace el término “ágil” aplicado al desarrollo de software. En esta reunión participan un grupo de diecisiete expertos de la industria del software, incluyendo algunos de los creadores o impulsores de metodologías de software.

Su objetivo fue esbozar los valores y principios que deberían permitir a los equipos desarrollar software rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto. Se pretendía ofrecer una alternativa a los procesos de desarrollo de software tradicionales, caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas.

Tras esta reunión se creó The Agile Alliance, una organización sin fines de lucro, dedicada a promover los conceptos relacionados con el desarrollo ágil de software y ayudar a las organizaciones para que adopten dichos conceptos. El punto de partida fue el Manifiesto Ágil, un documento que resume la filosofía “ágil”.

Valores de la Metodología Ágil

Según el Manifiesto se valora:

- **Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas**. La gente es el principal factor de éxito de un proyecto software. Es más importante construir un buen equipo que construir el entorno. Muchas veces se comete el error de construir primero el entorno y esperar que el equipo se adapte automáticamente. Es mejor crear el equipo y que éste configure su propio entorno de desarrollo en base a sus necesidades.
- **Desarrollar software que funciona más que conseguir una buena documentación**. La regla a seguir es "no producir documentos a menos que sean necesarios de forma inmediata para tomar un decisión importante". Estos documentos deben ser cortos y centrarse en lo fundamental.
- **La colaboración con el cliente más que la negociación de un contrato**. Se propone que exista una interacción constante entre el cliente y el equipo de desarrollo. Esta colaboración entre ambos será la que marque la marcha del proyecto y asegure su éxito.
- **Responder a los cambios más que seguir estrictamente un plan**. La habilidad de responder a los cambios que puedan surgir a lo largo del proyecto (cambios en los requisitos, en la tecnología, en el equipo, etc.) determina también el éxito o fracaso del mismo. Por lo tanto, la planificación no debe ser estricta sino flexible y abierta. [3]

Principios de la Metodología Ágil

Los valores anteriores inspiran los doce principios del manifiesto. Los principios son características que diferencian un proceso ágil de uno tradicional. Los dos primeros principios son generales y resumen gran parte del espíritu ágil. Los principios restantes tienen que ver con el proceso a seguir y con el equipo de desarrollo, en cuanto a metas a seguir y a organización del mismo. Los principios son:

I La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software que le aporte un valor.

II Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva.

III Entregar frecuentemente software que funcione desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas.

IV La gente del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto.

V Construir el proyecto en torno a individuos motivados. Darles el entorno y el apoyo que necesitan y confiar en ellos para conseguir finalizar el trabajo.

VI El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.

VII El software que funciona es la medida principal de progreso.

VIII Los procesos ágiles promueven un desarrollo sostenible. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener una paz constante.

IX La atención continua a la calidad técnica y al buen diseño mejora la agilidad.

X La simplicidad es esencial.

XI Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos.

XII En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto, ajusta su comportamiento. [3]

Comparación entre Metodologías Ágiles y Tradicionales (No Ágiles)

En la Tabla 1 se presentan las principales diferencias entre las metodologías ágiles y las tradicionales ("no ágiles"). Estas diferencias afectan no sólo al proceso en sí, sino también al contexto del equipo, así como a su organización.

Metodologías Ágiles	**Metodologías Tradicionales**
Basadas en heurísticas provenientes de prácticas de producción de código	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
Especialmente preparados para cambios durante el proyecto	Cierta resistencia a los cambios
Impuestas internamente (por el equipo)	Impuestas externamente
Proceso menos controlado, con pocos principios	Proceso mucho más controlado, con numerosas políticas/normas
No existe contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado
El cliente es parte del equipo de desarrollo	El cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio	Grupos grandes y posiblemente distribuidos
Pocos artefactos	Más artefactos
Pocos roles	Más roles
Menos énfasis en la arquitectura del software	La arquitectura del software es esencial y se expresa mediante modelos

Tabla 1 Diferencias entre metodologías ágiles y no ágiles [1]

Caracterización de las Metodologías Ágiles

No existe una metodología universal para hacer frente con éxito a cualquier proyecto de desarrollo de software. Toda metodología debe ser adaptada al contexto del proyecto (recursos técnicos y humanos, tiempo de desarrollo, tipo de sistema, etc.). Históricamente, las metodologías tradicionales han intentado abordar la mayor cantidad de situaciones de contexto del proyecto, exigiendo un esfuerzo considerable para ser adaptadas, sobre todo en proyectos pequeños y con requisitos muy cambiantes. Las metodologías ágiles ofrecen una solución casi a medida para una gran cantidad de proyectos que tienen estas características. Una de las cualidades más destacables en una metodología ágil es su sencillez, tanto en su aprendizaje como en su aplicación, reduciéndose así los costos de implantación en un equipo de desarrollo. Esto ha llevado hacia un interés creciente en las metodologías ágiles. Sin embargo, hay que tener presente una serie de inconvenientes y restricciones para su aplicación, tales como: están dirigidas a equipos pequeños o medianos (Beck sugiere que el tamaño de los equipos se limite de 3 a 20 como máximo, otros dicen no más de 10 participantes), el entorno físico debe ser un ambiente que permita la comunicación y colaboración entre todos los miembros del equipo durante todo el tiempo, cualquier resistencia del cliente o del equipo de desarrollo hacia las prácticas y principios puede llevar al proceso al fracaso

(el clima de trabajo, la colaboración y la relación contractual son claves), el uso de tecnologías que no tengan un ciclo rápido de realimentación o que no soporten fácilmente el cambio, etc.

Falta aún un cuerpo de conocimiento consensuado respecto de los aspectos teóricos y prácticos de la utilización de metodologías ágiles, así como una mayor consolidación de los resultados de aplicación. La actividad de investigación está orientada hacia líneas tales como: métricas y evaluación del proceso, herramientas específicas para apoyar prácticas ágiles, aspectos humanos y de trabajo en equipo. [1]

Aunque los creadores e impulsores de las metodologías ágiles más populares han suscrito el manifiesto ágil y coinciden con los principios enunciados anteriormente, cada metodología tiene características propias y hace hincapié en algunos aspectos más específicos. Entre éstas se pueden mencionar Programación Extrema (Extreme Programming, Xp, de Kent Beck) [2] [4], SCRUM5 (Ken Schwaber, Jeff Sutherland y Mike Beedle) [2] [4], Crystal Methodologies (Alistair Cockburn) [2] [4], Dynamic Systems Development Method (DSDM) [2], Adaptive Software Development (ASD, de Jim Highsmith) [2] [4], Feature-Driven Development (FDD, Jeff De Luca y Peter Coad) [2], Lean Development (LD, de Bob Charette).

Un método de desarrollo de software es ágil cuando es **incremental** (entregas pequeñas de software con ciclos rápidos), **cooperativo** (cliente y desarrolladores trabajan constantemente juntos en estrecha colaboración), **sencillo** (el método es fácil de aprender y de modificar) y **adaptativo** (capaz de realizar cambios a último momento). [2]

Propuesta Metodológica

Se propone un proceso de desarrollo ágil basado en la *prototipación evolutiva*, con ciclos conformados por la especificación o evolución de requerimientos, la especificación o evolución de diseño y la codificación. Para las especificaciones se considera el uso de sólo dos modelos: uno para representar estructuras de datos y otro para representar interfaces y funciones.

Para modelar datos se propone emplear diagramas de clase que representen el modelo de dominio *(entity classes)*, y para modelar interfaces y funciones se propone un modelo basado en la teoría de autómatas finitos: se concibe a todo sistema como a un autómata cuyos estados se asimilan a interfaces *(boundary classes)*, y sus transiciones a funciones (métodos de un *controller* asociado a cada interfaz) y que son la base para la codificación. Para cada ciclo de evolución se contemplan las etapas del proceso refinándose estos diagramas.

Para la *especificación de requerimientos* se clasifican los objetos entidad y se denotan sus relaciones, sin considerar aún sus propiedades o atributos en detalle, en el modelo de datos, y se especifica una primera aproximación funcional del sistema, considerando los actores o categorías de usuarios y/o la organización funcional del sistema de información (agrupamientos funcionales o interfaces tipo menú), en el modelo dinámico.

La *especificación de diseño* se efectúa refinando el modelo de datos mediante la especificación o agregado de atributos, y refinando el modelo dinámico mediante la especificación detallada de las interfaces, subdiagramas y transiciones.

Se trabaja en la definición de métricas de calidad, tanto para las estructuras de datos como para las funciones, para validar y orientar el diseño, y métricas de complejidad para la valorización de los sistemas que se desarrollen. También se está implementando una herramienta CASE que soporta esta metodología.

La participación y el compromiso de los usuarios finales en desarrollos basados en esta metodología se presumen garantizados debido a que los modelos empleados para las especificaciones son de un alto nivel de abstracción y comprensibles para personas no especializadas; además el modelo de interfaces y funciones, tal como el de casos de uso en el Proceso Unificado de Desarrollo permite verificar la completitud y rastrear el cumplimiento de requerimientos, con la posibilidad de la

prototipación temprana, asequible a partir de las mismas herramientas de desarrollo a partir de la especificación del diseño de interfaces, optimiza las relaciones contractuales facilitando la aprobación de fases y ciclos de evolución.

Referencias

[1] Canós, J., Letelier, P., Penadés, C. *Metodologías Ágiles en el Desarrollo de Software*. VIII Jornadas de Ingeniería del Software y Bases de Datos. 12-14 de Noviembre. 2003. Alicante, España. http://issi.dsic.upv.es/tallerma

[2] Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J. *Agile software development methods. Review and analysis*. Espoo 2002. VTT Publications 478. Finland. www.agilealliance.com/articles/index

[3] Beck, K. et al. *Manifesto for Agile Software Development*. Febrero. 2001. Utah, Estados Unidos. www.agilemanifesto.org

[4] Rising, L. Agile Methods: What's it All About? DDC-I Online News. 1 de Noviembre. 2001. www.agilealliance.com/articles/index