Ejecución de proyectos al estilo XP

IIC2143: Ingeniería de Software

Yadran Eterovic (yadran@ing.puc.cl)

2-2018

XP es una metodología ágil orientada a armar equipos cohesivos que se comunican bien,

… y crear un ambiente relajado y energizado

Cuando los equipos construyen código que es simple, no complejo, pueden acoger el cambio en lugar de temerle

Los equipos de software son exitosos cuando producen código de excelente calidad

Pero incluso los equipos realmente buenos, con desarrolladores muy talentosos, tienen problemas con su código:

- cuando cambios pequeños en el código se convierten en una serie de “hacks” en cascada
- o cuando los *commits* diarios significan horas arreglando conflictos por mezcla
- … el trabajo que solía ser gratificante se vuelve molesto, tedioso y frustrante

Aquí es donde entra XP

XP: Una mentalidad que ayuda al equipo y al código

Metodología ágil creada por Ken Beck y Ron Jeffries

Popular desde mediados de los 1990s

Enfocada no solo en la gestión del proyecto

… sino también en cómo los equipos realmente construyen código

…

…

Tiene prácticas y valores que ayudan a los equipo a desarrollar una mentalidad eficaz

… que, a su vez, ayuda a todos los participantes:

- a ser más cohesivos
- a comunicarse mejor
- a planificar mejor

… lo que permite suficiente tiempo para escribir bien el código

El desarrollo iterativo ayuda al equipo a estar en control de los cambios

Los equipos XP siguen el segundo principio del Manifiesto Ágil:

> "Acogemos requisitos cambiantes, incluso tarde en el desarrollo. Los procesos ágiles aprovechan el cambio para la ventaja competitiva del cliente"

XP es una metodología iterativa e incremental

… que emplea prácticas familiares, si ya conoces Scrum

- relatos, ciclo trimestral, ciclo semanal

Emplean relatos de usuario para seguirle la pista a los requisitos:

- normalmente los estiman en *horas de trabajo* necesarias para completarlos

Planifican su trabajo trimestralmente (*quarterly*, práctica muy "gringa"):

- se juntan y reflexionan sobre lo que pasó el trimestre anterior
- hablan sobre el foco de la empresa y el lugar que ocupa el equipo en ella
- planifican los temas* para el trimestre para seguirle la pista a sus objetivos de largo plazo
- planifican el backlog para el trimestre reuniéndose con usuarios e interesados

*Tema: Tal como un "objetivo de sprint" en Scrum, para asegurarse de que no pierdan de vista el *big picture*, es una o dos frases que describen lo que quieren lograr

Usan iteraciones de una semana —el ciclo semanal:

- empiezan con una reunión en que revisan el progreso hasta el momento

  … hacen una demo de exactamente lo que hicieron la semana anterior

  … trabajan con el cliente para elegir los relatos de esta semana,

  … y descomponen los relatos en tareas
- el ciclo empieza en martes o miércoles (todas las semanas el mismo día y a la misma hora), para evitar la presión de trabajar en fin de semana
- en la reunión inicial participa el cliente, para ayudar al equipo a elegir los relatos y estar al tanto del progreso

Cuando hacen un plan, agregan holguras (*slack*):

- algunos ítemes opcionales o de importancia menor (relatos “nice to have”) que pueden “botar” si se están atrasando
- … y que en total no ocupan más de un día de trabajo (de los cinco de la semana)

Dos valores de XP: La valentía y el respeto —dejan al miedo fuera del proyecto

XP depende de que el equipo tenga la mentalidad adecuada

Valentía:

- … para aceptar desafíos —a nivel de equipo
- … para apoyar o defender al proyecto —a nivel individual; p.ej., no comprometernos a un plazo que no podemos cumplir

…

…

Respeto:

- los integrantes del equipo se respetan mutuamente (lo que facilita la valentía) — p.ej., toman de verdad en cuenta ideas y opiniones, expresadas por otros integrantes, que pueden no gustarles
- todos confían en que cada uno haga su trabajo

XP y Scrum se enfocan en aspectos diferentes del desarrollo de software

XP no tiene el mismo nivel de focalización en gestión de proyecto

La “P” significa programación:

- XP está optimizado para ayudar a un equipo de programadores a mejorar la forma en que trabajan
- … así, está enfocado en desarrollo de software
- por supuesto, incluye suficiente gestión de proyecto como para poder hacer el trabajo

Lo que los une son ideas comunes y valores compartidos —Manifiesto Ágil

… por lo que muchos equipos ágiles usan una combinación de Scrum y XP

Un método híbrido a partir de XP y Scrum —práctica común— rompe las reglas de (al menos) uno de ellos

… pero está OK:

- las reglas de un método están para ayudarnos a ejecutar bien los proyectos

Los problemas ocurren si eliminamos o cambiamos un elemento que constituye un pilar de la metodolgía —y que a alguien le pareciera algo menor:

- p.ej., reemplazar al Dueño del Producto por un comité

…

…

En cambio, reemplazar las prácticas de ciclo semanal, ciclo trimestral, y holgura de XP con una implementación completa y sin cambios de Scrum, funciona:

- solo impacta la parte de planificación de XP
- … y no elimina ninguno de los pilares que hacen que XP funcione
- … por lo que a muchos equipos les ha ido bien haciéndolo

La práctica del *equipo como un todo*

Todos funcionan juntos como un equipo:

- cuando las personas trabajan juntas, se escuchan y se ayudan unos a otros a resolver sus problemas
- … escriben (mucho) más código
- … y el código que escriben es de (mucha) mejor calidad

Hay que hacer todos los esfuerzos necesarios para que las personas sientan que pertenecen

… y hay que ayudar a cada persona a apoyar a los otros integrantes del equipo

El equipo como un todo se construye a partir de la confianza:

- cuando encuentran obstáculos, todos trabajan juntos para superarlos
- cuando enfrentan una decisión importante, la toman entre todos

La confianza significa que tus compañeros pueden cometer errores:

- todos entendemos que se cometen errores
- … y que la única forma de progresar es cometer los errores inevitables y aprender de ellos todos juntos

No hay roles fijos:

- todos hacen un poco de todo —hablar con los usuarios, escribir relatos, escribir código, diseñar la interfaz de usuario, diseñar la arquitectura, gestionar el proyecto
- los roles cambian dependiendo de las habilidades de cada uno —cuando hay que hacer un trabajo, el equipo se asegura de que la persona apropiada lo haga

La práctica de *sentarse juntos*

La programación es una actividad muy social:

> "El método más eficiente y eficaz de transmitir información hacia y dentro de un equipo de desarrollo es la conversación cara a cara"

Cuando estás en un equipo de desarrollo, necesitas información todo el tiempo:

- qué estamos desarrollando
- cómo vamos a construirlo
- cómo encaja lo que yo estoy haciendo en el resto del software

…

…

=> muchas conversaciones cara a cara

Los equipos trabajan mejor, e innovan, cuando se sientan juntos:

- en la misma parte de la oficina
- … de modo que sea fácil y conveniente para cada uno encontrar a la persona que necesitan y tener una conversación cara a cara

Los equipos XP valoran la comunicación

Las personas en el equipo trabajan juntas:

- planifican juntas
- colaboran para entender lo que van a construir
- escriben código juntas

Si uno pertenece a un equipo XP,

… uno realmente cree que cuando tiene un problema

… va a encontrar la mejor solución si se comunica con sus compañeros

La programación es un trabajo creativo e intelectual

… y ponen en práctica el *lugar de trabajo informativo*

Las personas están absorbiendo información constantemente de lo que está y de los que están alrededor de ellas:

- usan herramientas visuales, p.ej., el tablero de tareas o el diagrama *burndown*
- se sientan cerca de personas que están hablando acerca del proyecto —se pueden desconectar cuando quieran, p.ej., poniéndose *headphones*

La práctica de *trabajo energizado*

Los equipos de software tienen que innovar todo el tiempo:

- diseñar nuevos productos, implementar nuevas ideas, entender lo que la gente necesita, resolver problemas lógicos complejos, verificar y validar lo que uno ha hecho

Se requiere una mente relajada y descansada

Hay que darse el tiempo suficiente para hacer el trabajo:

- los plazos irreales son la manera más fácil de destruir la productividad de un equipo, su moral, y la alegría de hacer el trabajo

Hay que eliminar las interrupciones:

- silenciar los teléfonos y apagar las notificaciones de email por un par de horas al día facilita llegar a ese estado de concentración profunda en el que uno casi no nota el paso del tiempo

Hay que permitirse cometer errores:

- construir software es innovar constantemente

  … y el fracaso es el fundamento de la innovación
- hay que verlo como experiencia de aprendizaje

Hay que trabajar a un ritmo sostenible:

- los equipos que trbajan muchísimas horas todos los días e incluso los fines de semana producen código de peor calidad y finalmente producen menos código

La personas en un equipo XP toman muy en serio el balance entre trabajo y vida personal

Se van a casa a una hora razonable todos los días

… y tienen vidas y familias fuera del trabajo

La realimentación frecuente ayuda a que los cambios sean pequeños, y tiene muchas formas

Es muy raro que un equipo produzca el código correcto al primer intento

Iteraciones: El equipo produce una parte pequeña del código y recibe realimentación de los usuarios —nueva información que se usa para mejorar cómo se planifica la próxima iteración

Integración de código: Al integrar frecuentemente tu código con el de tus compañeros, recibes realimentación temprana

Revisiones por compañeros de equipo: … que además los ayuda a ellos a entender lo que tú produjiste

Pruebas unitarias: Cuando haces un cambio a tu código y ahora éste no pasa una prueba, es una de las realimentaciones más valiosas que puedes tener

Las prácticas XP nos dan realimentación sobre el código desde temprano y a menudo

Se producen ciclos de realimentación:

- el equipo aprende de cada rueda de realimentación, y hace ajustes y se corrige a sí mismo, lo cual cambia lo que el equipo va a aprender en la próxima rueda

Un sistema que da mucha realimentación a menudo falla rápido:

- queremos que nuestro sistema falle rápidamente para poder corregir los problemas temprano, antes de agregar otras partes del sistema

Las prácticas:

- programación en pares
- *builds* (compilación del código fuente, *testing* y empaquetamiento del código ejecutable) que toman 10 minutos
- integración continua
- desarrollo dirigido por pruebas

Los equipos XP usan *builds* automatizados que corren rápido

Cuando el *build* requiere mucho trabajo manual o toma más de 10 minutos en ejecutarse,

… tensiona al equipo —los programadores son impacientes y los frusta esperar— y hace que el proyecto avance más lentamente

…

…

Los equipos usan herramientas o lenguajes de *scripting* especializados

… para crear un *build* automatizado al inicio del proyecto

La clave es asegurarse de que el build corra en menos de 10 minutos:

- límite de la paciencia que los programadores tienen para esperar que el *build* termine
- ayuda a encontrar problemas con el *build* rápidamente
- tiempo suficiente para tomarse un café y relajar el cerebro

La integración continua —varias veces al día— previene sorpresas desagradables

Si el equipo no usa un sistema de control de versiones, sino que simplemente una carpeta compartida, entonces rápidamente se va a enfrentar a un gran enredo

Si el equipo usa un sistema de control de versiones —con protocolos para hacer *check* out y *check in* de código— entonces el sistema se encarga de mezclar los cambios correcta y automáticamente

…

… a menos que se trate de cambios que entran en conflicto entre sí —cambios ingresados recientemente en la misma línea del mismo archivo por alguien más:

- el sistema rechaza tu cambio, pero actualiza su versión local con ambos cambios
- … y tú eres responsable de resolver el conflicto (el sistema no tiene cómo)

Si hay muchos conflictos, porque alguien esperó un mes antes de hacer un solo check in de todos sus cambios, entonces resolverlos se vuelve muy difícil

El ciclo semanal comienza escribiendo las pruebas (*tests*) … sí, antes de escribir el código mismo

Los equipos XP emplean la práctica de *desarrollo dirigido por pruebas*, o TDD:

- los programadores escriben pruebas unitarias (*unit tests*) antes de escribir el código que las pruebas van a probar
- … lo que te obliga a pensar qué significa que tu código funcione correctamente
- … lo que te ayuda a escribir código "realmente" terminado
- se repite cíclicamente, funcionalidad por funcionalidad

…

…

Las pruebas unitarias son pruebas que prueban unidades de código:

- clases, funciones, etc., dependiendo del lenguaje de programación
- se escriben en el mismo lenguaje que el resto del código

Ayuda a evitar problemas de diseño del código que se manifiestan la primera vez que escribes código adicional que hace uso del anterior:

- al escribir las pruebas primero, tú en efecto *usas* el código que vas a escribir a continuación

Los equipos ágiles obtienen realimentación del diseño y de las pruebas (*testing*)

Usan “herramientas” de diseño y de testing que les permiten recibir más realimentación a lo largo del proyecto

wireframes: desde bosquejos de la navegación hasta representaciones detalladas de páginas, para recibir realimentación sobre la interfaz de usuario

soluciones *spike*: código escrito específicamente para descubrir las causas y aprender más de problemas técnicos difíciles

pruebas de usabilidad: para asegurarse de que han tomado buenas decisiones de diseño

pruebas exploratorias: encontrar nuevas combinaciones de acciones en las que el desarrollador no pensó

Programación en pares

Dos personas se sientan frente a un mismo computador y escriben código juntas

El equipo rota constantemente los pares,

… de modo que todos ganan experiencia trabajando con cada parte del sistema

Producen más que si trabajan por separado:

- se ayudan a mantenerse enfocados
- conversan constantemente sobre los problemas en los que están trabajando y van proponiendo una solución tras otra
- al explicarle tu idea a la otra persona, te das cuenta de si la entiendes bien
- cuando uno no sabe cómo seguir, el otro puede ayudar
- es menos probable que uno use “atajos” si hay alguien más trabajando al lado
- hay dos pares de ojos observando cada cambio, por lo que encuentran muchos de esos pequeños errores que después pueden llegar a producir dolores de cabeza