Planeamiento de proyectos

IIC2143 : Ingeniería de Software

Yadran Eterovic (yadran@ing.puc.cl)

Los clientes quieren el software cuando lo necesitan

… en **90 días** y ni un día después

Tú has empezado a entender las ideas del cliente en una sesión de brainstorming

… tienes un conjunto de relatos de usuario que describen todo lo que el cliente quiere que el software haga

… e incluso has agregado una estimación a cada relato, que te ha permitido hacerte una idea de cuánto te vas a demorar para entregar todo lo que el cliente quiere:

489 días

El cliente confirma que quiere la aplicación, solo que no puede esperar casi dos años

Tú revisas los relatos de usuario y seleccionas un subconjunto que se podría hacer en 90 días

… pero al hablar con el cliente:

- “¡Pero esto no es ni cerca lo que yo quiero!”

¿Qué pasó?

Después de todo el trabajo para entender lo que el cliente quería, lo ignoramos a la hora de decidir cuáles relatos tienen prioridad

Al priorizar relatos de usuario, hay que mantenerse enfocados en el cliente:

- solo el cliente sabe lo que realmente se necesita
- … así que al decidir que va y que no
- … es una elección que el cliente tiene que hacer

Pidámosle al cliente que ordene los relatos por prioridad y que elija los que tienen que ser entregados en el Hito 1.0 (o Versión 1) del software

El **Hito 1.0** va a ser nuestro primer *major release* del software al cliente:

- luego de algunas iteraciones en que hemos recibido realimentación, aquí esperamos que el cliente nos pague

Hay que balancear la funcionalidad con la impaciencia del cliente:

- ayudemos al cliente a entender qué se puede hacer en el tiempo disponible
- los relatos que no van en el Hito 1.0, solo se posponen para el Hito 2, o 3…

No hay que quedarse atrapado planeando *nice-to-haves*:

- se trata de entregar lo que se necesita —la funcionalidad que cumple con las necesidades más importantes del cliente

Finalmente, tomamos todos los relatos que el cliente quiere que vayan en el Hito 1.0 y sumamos las estimaciones correspondientes:

273 días

Hay que repriorizar con el cliente

Hay que dejar fuera más funcionalidad, sacando relatos de usuario que no sean absolutamente cruciales

… para entregar **exactamente la funcionalidad que se necesita para una versión útil del software** —*funcionalidad línea de base*

¿Y si aún así no puedo entregar lo que el cliente quiere cuando lo quiere?

- seamos honestos con el cliente y retirémonos del proyecto
- … o tratemos de agregar más personas al equipo

Pero en último caso, no es tan simple como “el doble de personas, la mitad de la estimación”

Todo integrante nuevo del equipo tiene que ser puesto al día en el proyecto:

- entender el software, las decisiones técnicas, y cómo todo calza con todo

Hay que entregarle las herramientas (¿pagar licencias?) y el equipamiento (¿comprar?) necesarios:

- por lo menos, toma tiempo

…

…

Se hace más difícil mantener a todos focalizados y sabiendo qué están haciendo:

- esto se puede convertir en un trabajo de jornada completa y afectar la productividad del equipo

Hay un número máximo de personas que el equipo puede tener y seguir siendo productivo:

- depende del proyecto, del equipo y de la persona que estamos agregando
- hay que monitorear el equipo y asegurarse de que no baje la productividad

Así, agregar más personas al equipo no siempre funciona como se espera

Si el equipo crece hasta 4 personas, el desempeño puede mejorar hasta 4 veces

De 6 a 12 personas, el desempeño mejora, pero mucho menos que 6 a 12 veces, debido a la comunicación adicional

A partir de 15 personas, el desempeño empeora

En todo caso, no hay exactamente un tamaño máximo de equipo que funcione igual para todos los equipos

Supongamos que incorporas dos amigos a tu equipo (de uno),

…y que entre los tres pueden producir el equivalente a 190 días de trabajo en 90 días (sacado de alguna tabla “mágica”):

273 – 190 = 83 días de trabajo que todavía faltan

Hablamos una vez más con el cliente y sacamos algunos otros relatos:

llegamos a un **Hito 1.0 de 184 días**

A partir de este punto, tenemos algo que sí podemos terminar en el tiempo disponible:

- pidámosle al cliente que priorice los relatos que están en el Hito 1.0 —uno puede ayudar y aconsejar, p.ej., haciendo ver dependencias entre algunos relatos— pero **la decisión final sobre las prioridades es siempre del cliente**
- organicemos los relatos, priorizados por el cliente, en iteraciones

Apuntemos a iteraciones cortas:

- mientras más cortas, más rápido tienes realimentación, y más oportunidades hay de encontrar y hacernos cargo* de cambios y detalles inesperados a medida que aparecen
- *ajustar nuestros planes e incluso cambiar lo que estamos haciendo en la próxima iteración, antes de entregar un Hito 1.0 que no sirve

Apuntemos a iteraciones balanceadas:

- en términos de hacernos cargo de cambios, agregar nuevas funcionalidades, *debugging*,
- … y tomar en cuenta que trabajamos con personas de verdad —una iteración de 30 días calendario corresponde a unos 20 días de trabajo de desarrollo de software

Comparemos nuestro plan con la realidad:

- cita con el doctor, hoy llego a las 11, no a las 8.30
- la gente de informática va a estar instalando la nueva versión de Oracle en mi computador toda la tarde de hoy
- tengo una semana de vacaciones a fin de mes

Es decir, no siempre tenemos al 100% del equipo trabajando el 100% del tiempo

¿Qué podemos hacer?

La **velocidad** toma en cuenta el *overhead* en nuestras estimaciones —es el porcentaje (entre 0 y 1) de tiempo realmente productivo

Si no tenemos más información, empecemos con una velocidad de 0.7:

- sólo el 70% del tiempo disponible realmente lo destinamos al desarrollo del proyecto

Si a medida que pasa el tiempo nos damos cuenta de que la velocidad del equipo es otra,

…entonces, al final de la iteración vigente la ajustamos para planificar la próxima iteración

Los programadores piensan en días utópicos

Si le preguntamos a un programador cuánto se va a demorar en programar algo

… nos va a dar una estimación mejor que el mejor caso posible:

- “Ningún problema. Lo tengo listo en dos días.”

Pero … ¿qué está pensando realmente?

- supone que es la única persona involucrada, que no va a cometer errores, que el testing es responsabilidad de alguien más
- … que va a programar hasta las 3 am, que va a seguir trabajando en la mañana, que nadie lo va a molestar en todo el día, etc.

Los desarrolladores debemos pensar en días del mundo real

Debemos lidiar con la realidad:

- equipo de programadores, cliente

Partimos con una iteración de un mes (calendario),

… tomamos en cuenta fines de semana y feriados, y nos quedan 20 días,

… y aplicamos la velocidad de 0.7, para tomar en cuenta el tiempo que no dedicamos al desarrollo,

… y nos quedan 14 días de trabajo real

Incluyamos la velocidad *antes* de dividir en iteraciones

En una iteración de un mes calendario, un equipo de 3 desarrolladores puede producir

… 3 x 20 x 0.7 = 42 días de trabajo real

Por lo tanto, a lo largo de 3 iteraciones (90 días calendario) vamos a producir (solamente)

… 42 x 3 = 126 personas-días de trabajo

Hay que darle las malas noticias al cliente

Este es el momento más temido por todo desarrollador:

- planificamos las iteraciones
- … incluimos la velocidad de nuestro equipo
- … y no vamos a poder tener todo lo que el cliente quiere a tiempo

Pero no hay trucos mágicos

… tenemos que contarle al cliente y ver qué quiere que hagamos:

- el cliente: “¡Qué lata! Así que pueden hacerlo todo menos la funcionalidad de viajero frecuente. Hmm…déjenme pensarlo…”

¿Qué hacer con clientes enfurecidos?

Seamos honestos

Queremos producir un plan para el hito 1.0 que podamos cumplir

… no un plan que dice lo que el cliente quiere que diga (y que nos pone en un *fast track* al fracaso)

Prometamos y cumplamos

… en vez de prometimos demasiado y no cumplimos

Ofrezcamos opciones realistas y expliquemos

Agregar una iteración al hito 1.0:

- 42 x 4 = 168 personas-días … en 4 meses lo hacemos (casi) todo

Expliquemos que lo que no vamos a hacer en el hito 1.0 no está perdido, sólo pospuesto hasta el hito 2.0

Transparentemos cómo llegamos a los números—el cliente sólo tiene nuestra palabra de lo que podemos hacer:

- queremos entregarle software exitoso —tuivmos que sacrificar algunas funcionalidades
- … para hacer un plan en el que confiamos que podemos cumplir

Mantengámonos confiados en que vamos a lograr lo que prometimos

Confiemos en nuestros planes

… aplicando velocidad

… y no trabajando en exceso

El cliente: “OK. Para mí vale la pena perder la funcionalidad de viajero frecuente con tal de mantener las cosas andando. Hagámoslo.”