Contratos de Alcance Opcional

Kent Beck, First Class Software
Dave Cleal

"Ahora, los clientes pueden tener aquello que quieren al final del proyecto, luego de haber aprendido; en lugar de lo que querían al comienzo del proyecto."

1 Introducción

"Por favor, proponga un sistema que satisfaga la especificación de requerimientos adjunta. Especifique fecha de entrega y costo total. " Estas palabras son suficientes para introducir la codicia y el miedo en el corazón de los programadores. Codicia, porque si usted posee un perfil técnico y experiencia en el área del sistema propuesto, puede obtener ganancias muy superiores a las obtenidas a través de cualquier tarifa diaria concebible. Miedo, porque cuando algo anda mal en un contrato a precio fijo, anda decididamente mal. En el mejor de los casos usted puede esperar negociaciones tensas. En el peor: costos irrecuperables, pérdida de reputación o una demanda judicial.

Kent hace una digresión:

> Durante los buenos viejos días de la comercialización de Smalltalk, firmé un contrato para entregar un motor de planillas de cálculos a un banco de inversión de Wall Street. No estaba seguro sobre cuáles eran las implicaciones, pero estaba seguro de que podría hacerlo en seis semanas. Contraté a mi antiguo colaborador Ward Cunningham y nos refugiamos durante dos días para lograr un buen comienzo. Al final de los dos días, básicamente habíamos terminado. Pasé otros dos o tres días puliendo, holgazaneé durante cuatro semanas, y finalmente entregué el software con una semana de antelación. El cliente estuvo suficientemente contento como para firmar un contrato para desarrollar extensiones.
>
> Esta vez tenía una base. Conseguí otro programador para implementar los cambios, los cuales no eran tan complicados. Me mantuve en contacto con él día a día. Parecía estar haciendo buenos progresos. Sin embargo, una semana antes de la fecha de entrega, silencio ominoso. Dos días antes de la fecha de entrega, recibo una extraña llamada. "Emmmm... Estoy en el hospital. Mi apéndice se reventó. Estoy tan medicado que no puedo ver bien. Espero que no haya problema."
>
> Aaaaaaaah! Pánico. Tomé el código que había recibido hasta el momento, junté unas cuantas bebidas energizantes (tengo una reserva de bebidas energizantas para casos especiales de pánico), y en una maratónica sesión de programación de 48 horas logré cumplir con el plazo de entrega. No hace falta aclarar que la calidad de la segunda entrega no estuvo a la altura de la primera, y que el cliente no quedó satisfecho.

2 Contratos de Alcance Fijo

La historia anterior ilustra algunas de las ventajas y desventajas de los contratos de alcance fijo. En general, un contrato de precio/fecha/alcance fijo parece trasladar el riesgo del proyecto del cliente al proveedor. El riesgo más grande es que el software sea más difícil de implementar que lo esperado. Si el riesgo no se materializa, la apuesta del proveedor paga y ese tipo de contratos pueden resultar muy lucrativos. Sin embargo, si ocurre lo peor, el proveedor queda en posición de perder mucho dinero, ya sea por cancelar el contrato o por terminarlo dando pérdida.

Los contratos de precio/fecha/alcance fijo existen porque parecen servir a las necesidades tanto de clientes como proveedores. Parece que los clientes obtienen:

- Costo predecible
- Entragables predecibles
- Agenda predecible

Y los proveedores deberían obtener:

- Ganancias predecibles
- Exigencias predecibles

¿Qué es lo que está mal en este esquema? Para empezar, la única parte predecible de los costos del cliente es el importe que se le paga al proveedor. ¿Pero qué pasa cuando una característica del software resulta ser más difícil de implementar que lo esperado?

Puede que el cliente no tenga que pagarle nada más al proveedor, pero si el trabajo se entrega tarde o con bajos estándares de calidad, los costos indirectos pueden ser enormes. Incluso peor, ¿qué ocurre cuando alguien se da cuenta de que una característica es menos importante que lo que se creía, o las circunstancias cambian las prioridades que habían determinado el alcance original? Es demasiado complicado seguir renegociando el contrato, así que el problema es archivado hasta la versión 2.0 del sistema, mientras la versión 1.0 tiene características que todos saben que no satisfacen los requerimientos reales.

El proveedor puede tener exigencias predecibles, pero si su entendimiento de los requerimientos resulta ser erróneo, se enfrentan a una decisión poco envidiable. O agregan trabajo extra no remunerado o bien inician una áspera discusión sobre el significado detallado del contrato. En el mejor de los casos la discusión los deja con un cliente menos para el futuro. En el peor, llega la hora de sentarse con los abogados.

El problema básico es que un contrato de precio/fecha/alcance provoca una contraposición directa de los intereses del cliente y el proveedor. Considere las egoístas (léase "bajo estrés") posiciones de ambas partes:

Cliente	Proveedor
Interpreta los requerimientos tan ampliamente como sea posible, para obtener tanto por su dinero como sea posible.	Interpreta los requerimientos tan estrictamente como sea posible, para reducir recursos.
Quiere el trabajo terminado lo antes posible	Quiere terminar el trabajo sobre la fecha, para obtener seguir con el siguiente trabajo en la fila.
Quiere calidad superlativa.	Quiere invertir lo mínimo necesario en calidad de forma que el cliente pague.
Los programadores quemados por estrés no son problema del cliente, a menos que amenacen la entrega [1].	Quiere que los miembros del equipo tengan éxito en este proyecto, y que estén ahí para el siguiente.

Lo que necesitamos es una clase diferente de contrato, que encolumne las posiciones de ambas partes. Las cuatro variables de la gestión de proyectos son:

- Tiempo
- Costo
- Alcance

[1] En realidad, por supuesto, los programadores quemados por estrés son problema del cliente a largo plazo, porque su producto inmantenible es problema del cliente. Ocurre que cuando se está analizando un contrato es díficil pensar más allá de la finalización del mismo

- Calidad

El contrato de alcance fijo especifica valores para tres de estas variables:

- Tiempo
- Costo
- Alcance

En muchos contratos la otra variable, la calidad, ondea libremente en la brisa. Pero esto no tiene sentido alguno para el cliente, ya sea en el corto o largo plazo. Simplemente converse con los usuarios "reales" al filo del fin del acuerdo. En el corto plazo, ellos quieren algo que funcione, y lo quieren rápido. Encontrarán el modo de vivir sin ese complejo reporte o la función que se utiliza una vez al mes. No es que lo tengan ahora. Sin embargo, en el largo plazo necesitarán cambios. Más les vale que el software sea mantenible.

3 Alcance y Calidad Fijos

¿Por qué no especificar las cuatro variables? De hecho, esa es la forma en la que muchos proyectos comienzan. Los programadores en realidad prefieren hacer trabajos de alta calidad, por lo que tienden a establecer algunos estándares (usualmente bastante altos). Lo primero que ocurre a medida de que los inevitables conflictos surgen es que los programadores empiezan a trabajar más duro. Los líderes de proyecto mediocres aplauden a los programadores que hacen esto. Los líderes de proyecto más capaces ven este comportamiento como una señal temprana de problemas en el porvenir. Eventualmente los límites humanos de resistencia se hacen notar, y algo tiene que ceder. ¿Le dice el programador a su líder que no van a llegar a tiempo a la fecha de entrega del viernes? No, porque hay muchas formas de llegar a una fecha demasiado apretada:

- decidir que el software está complete incluso cuando se sabe que no maneja todos los casos
- dejar de desperdiciar tiempo (sic) testeando
- entregar la primera versión que funciona en lugar de mejorar su diseño para hacerla mantenible o incluso comprensible
- trabajar toda la noche hasta tener dificultades en la vista, y esperar lo mejor

Cuando las cuatro variables están especificadas, la calidad es la primera que se resigna, porque es la menos visible - o es la menos visible antes de que el software sea entregado.

4 Contratos de Alcance Opcional

Si fijar el tiempo, el alcance y el costo no funciona, y fijar las cuatro variables tampoco, ¿qué tal si planteamos alguna otra combinación? ¿Qué tal si fijamos:

- Tiempo
- Costo
- Calidad

y dejamos que el alcance absorba la incertidumbre? ¿Y si hacemos que el contrato sea realmente corto? El contrato luciría así:

> Pagaremos por el equipo de seis $75.000 por mes, por los próximos 2 meses. Cualquiera sea el software que entreguen, cumplirá los estándares de calidad especificados en letra chica más abajo. Hay algunas estimaciones iniciales en el apéndice A, pero son sólo por diversión.

Esto no puede funcionar. El cliente no sabe qué obtendrá a cambio de sus $150.000. Los proveedores podrían holgazanear por 2 meses y entregar una pantalla de login de muy alta calidad.

Primero, eso no ocurrirá. El proveedor quiere repetir el negocio. Si el proyecto completo va a durar un año, el proveedor sólo tiene 1/6 del dinero luego del primer contrato. En cualquier caso, mantuvimos el contrato corto, así que el cliente sólo arriesga el dinero correspondiente a 2 meses antes de determinar si el progreso es razonablemente rápido[2]. Las estimaciones iniciales para las características y un poco de matemática le darán al cliente una idea bastante aproximada de lo que podrían obtener en los próximos contratos.

Segundo, ¿qué pasa cuando el entendimiento actual de los requerimientos resulta erróneo? Ambas partes simplemente cambian de dirección. El proveedor no tiene motivación para no responder, porque descartar software existente no afecta su capacidad de cumplir el contrato.

Revisemos aquellos intereses contrapuestos.

Cliente	Proveedor
Interpreta los requerimientos tan ampliamente como sea posible, para obtener tanto por su dinero como sea posible.	Está feliz de aceptar los cambios de interpretación de requerimientos del cliente.
Quiere el trabajo terminado lo antes posible	Quiere entregar a tiempo. Feliz de entregar toda la funcionalidad que sea posible, sin arriesgar por eso la calidad.
Quiere calidad superlativa.	Quiere calidad antes que cualquier otra cosa.
El cliente quiere que los programadores quieran volver por los próximos 2 meses.	Quiere que los miembros del equipo tengan éxito en este proyecto, y que estén ahí para el siguiente. Si se trata de un proyecto de un año de duración, el proveedor sólo tiene 1/6 del dinero luego del primer contrato.

Vemos más intereses en común. El conflicto más probable ahora es que el proveedor insista en mantener el nivel de calidad mientras que el cliente ansioso trata de recortar detalles para embutir una característica más.

El contrato de alcance fijo tenía las siguientes ventajas:

Los clientes obtienen:

- Costo predecible
- Entregables predecibles
- Agenda predecible

Los proveedores obtienen:

- Ganancias predecibles
- Exigencias predecibles

El contrato de alcance opcional conserva todas estas ventajas, excepto la de "entregables predecibles". El cliente tiene una idea razonable de qué puede obtener al principio, pero la realidad se entromete rápidamente. Lo que obtendrá es inevitablemente diferente de lo que imaginaron al principio. Al resignar la ilusión de control sobre el alcance al principio del contrato, el cliente obtiene algo mucho más valioso. En lugar de obtener lo que quería al principio del proyecto, ahora puede tener lo que quiere al final del proyecto, luego

[2]De hecho usted arriesga menos que eso. Un contrato de alcance fijo podría dejarlo con un sistema enorme pero inusable (y en última instancia, sin valor alguno). Al menos esta pantalla de login tendrá atributos de calidad como para ser una base potencial para un sistema valioso.

de completar ese proceso de aprendizaje. Lo que aprende a querer es inevitablemente diferente de lo que *podría* haber imaginado al principio del contrato.

Más importante, sin embargo, es que el contrato de alcance opcional hace de la calidad una constante. El equipo trabajará, por un tiempo y precio fijos, al nivel de calidad exigido por el cliente. Nada estará "80% completo". Puede que sólo el 80% de las características sean completadas, pero cada una de ellas estará 100% completa, así como demostradas por tests automatizados.

Una característica importante de los contratos de alcance fijo no se mantiene en los contratos de alcance opcional. El proveedor no puede terminar un trabajo en un tercio del tiempo y cobrar el total del valor (las cuatro semanas de holgazanería en la historia introductoria de Kent). Si el proveedor termina antes de tiempo, pide más trabajo. Sin embargo, los contratos de alcance opcional son más valiosos para los clientes. En los casos en que hemos firmado contratos de alcance opcional, el valor agregado extra que brindamos nos ha permitido incrementar lo cobrado lo suficiente como para dejar pasar con gusto las posibilidades de beneficios inesperados.

5 Registrando requerimientos

Ok, imaginemos que les hemos vendido este enfoue a nuestros clientes y proveedores. Ahora están comprometidos por un contrato que excluye los requerimientos funcionales. Entonces, ¿qué van a desarrollar los proveedores? Obviamente aún necesitamos alguna forma de registrar requerimientos. ¿Cuál es la forma apropiada de hacerlo?

Un requisito particular de este contrato es que no esperemos que el cliente entregue "los requerimientos" alguna vez. Obtendremos suficientes requerimientos para empezar, y luego más a medida que vayamos aprendiendo. Entonces, necesitamos que nuestros requerimientos sean registrados en trozos del tamaño de un bocado, y necesitamos que el cliente priorice esos trozos.

La lista priorizada puede ser registrada de diversas maneras, pero cada registro debe describir algo por lo que el cliente está dispuesto a pagar. Algunas personas lo denominan "caso de uso"; otros lo llaman "un conjunto de casos de uso"; otros lo llaman una "historia". Nosotros utilizaremos la palabra "historia". Lo importante es que:

- Los programadores deben ser capaces de imaginar cómo pueden verificar que el sistema realmente implemente la historia. Los casos de test automatizados son un mecanismo simple y objetivo para esta verificación.
- Los programadores deben ser capaces de estimar cuánto esfuerzo requerirá esta historia, en términos relativos a las otras historias. Sin estimaciones, el cliente no puede tomar decisiones informadas de priorización.
- La historia debe ser comprensible para los usuarios del sistema. Si el cliente va a comparar la prioridad de esta historia con otras, deben tener una noción tanto de su valor como de su costo.
- Necesitamos implementar más o menos historias para usar el tiempo disponible. Por eso son mejores las historias más chicas, porque permiten que los desarrolladores trabajen en paralelo, y nos permiten utilizar todo el tiempo disponible. Nuestro contrato fuerza a los proveedores a entregar sólo software de alta calidad. Las historias parcialmente completas no estarán en el software entregado: por eso una historia muy grande tiene el riesgo de que el cliente no obtenga nada luego de una gran inversión o esfuerzo.

Hay una tendencia en la ingeniería de software a intentar fijar tanta información como sea posible antes de implementar, para evitar costos altos al final del proceso. Si podemos de alguna forma mantener la flexibilidad del software, sin embargo, podemos un vistazo mucho más rápido desde 30.000 pies de altura, antes de lanzarnos en paracaídas hacia la acción.

En particular, la cantidad de información que usted necesita sobre nuestros "trozos" de funcionalidad antes de que podamos estimarlos o priorizarlos es mucho menor que la cantidad de información que necesitamos para realmente implementarlos. Esto se debe a que nuestro contrato no nos lleva a tratar estimaciones "erróneas" como un problema grave. Podemos esperar que la primera parte del desarrollo, antes de que el cliente nos haya ayudado a enfocarnos al establecer prioridades, pase rápidamente.

6 Conclusión

Estas son las características que hacen que un contrato de alcance opcional sea valioso:

- Los clientes pueden cambiar de idea
- Los proveedores no son llevados a sacrificar la calidad tan pronto como algo sale mal
- Los intereses de clientes y proveedores están contractualmente alineados
- El conocimiento que ambas partes obtienen *durante* el proyecto puede influír en el producto terminado.

Todas estas características giran alrededor de la gestión del cambio. Imagine un proyecto donde el cliente sabe todo sobre el problema desde el principio, el proveedor ha desarrollado una docena de proyectos similares antes, y ambos lados están seguros de que no habrá sorpresas durante el proyecto ni aprenderán nada durante el mismo. Ese tipo de proyectos no ganarán nada de este tipo de organización, y los modelos tradicionales estarán bien.

Pero si no es ese el caso, algo inesperado va a ocurrir durante el proyecto. Cuando eso ocurra, usted quiere que el cliente y el proveedor trabajen juntos para lidiar con el problema, y quiere ser capaz de cambiar la forma del proyecto de forma tal que refleje la nueva realidad. En ese caso, considere un contrato de alcance opcional.

Incluso más radicalmente, un contrato de alcance opcional significa que usted no necesita siquiera una opinión sobre ciertas cosas. ¿Cuando terminamos? Después de tantos ciclos como querramos. ¿Qué queremos en última instancia del sistema? No estamos seguros: lidiaremos con eso luego de que las cosas obvias hayan sido implementadas. El contrato de alcance opcional le permite al equipo realizar progresos útiles y al mismo tiempo diferir decisiones sobre cosas que todavía no entiende muy bien.

Las implicaciones de desarrollar un contrato de alcance opcional desafían las estrategias de desarrollo de software convencionales.

- Diseño evolutivo - Usted no quiere invertir en funcionalidades que no implementará, por eso no puede analizar y diseñar el sistema de una sola vez. Debe estar preparado para evolucionar el análisis y el diseño a través del desarrollo.
- Entrega evolutiva - Usted quiere hacer dinero con las funcionalidades que ya han sido implementadas mientras espera que las funcionalidades "opcionales" se clarifiquen a sí mismas. Debe poner un sistema minimal en producción, y seguir implementando funcionalidades mientras soporta el sistema actual.
- Calidad interna y externa - Usted espera cambiar el sistema, por eso debe invertir en la calidad que le permita replantear radicalmente el sistema a través del proyecto sin costos prohibitivos para retrabajar y retestear.
- Automated testing - Cuando termina de implementar una funcionalidad, usted debe demostrar que está terminada, y quiere asegurarse de que no se rompa en el futuro. Los tests automatizados demuestran las funcionalidades.

Quizás la barrera más alta a la aceptación de los contratos de alcance opcional es que los programadores no pueden creer que los clientes aceptarían contratos así. Si los clientes están felices con su relación con sus proveedores, esto probablemente sea verdad, y ciertamente debería serlo. Sin embargo, si los clientes no

están contentos y los programadores tampoco, algo tiene que cambiar. Los contratos de alcance opcional ganan fuerza precisamente en esas circunstancias que son más plausibles de ocasionar problemas en casos de contratos de alcance fijo: requerimientos vagos y cambiantes.

6.1 Definiciones contractuales de calidad

Claramente este estilo de contrato sólo es tan efectivo como su definición de calidad. Una revisión completa de cómo definir buenas métricas de calidad sería el objeto de otro artículo, mucho más largo. Sin embargo, aquí hay algunos punteros a formas en las que creemos que la calidad puede ser controlada.

La mayoría de las métricas de calidad tradicionales se basan en la observación externa del software. ¿Está suficientemente libre de bugs? Simplemente cuente los bugs reportados a lo largo de tres meses y compárelos con algún número objetivo máximo de bugs. El problema con esta clase de enfoque es que para el momento en el que usted se da cuenta de que la calidad del software es pobre, ya le ha pagado al proveedor, los desarrolladores se desperdigaron a los cuatro vientos, y es simplemente muy tarde para hacer algo al respecto.

Entonces, en vez de hacer eso, buscamos métricas de calidad interna. Como argumentamos en este artículo, creemos que un elemento esencial de la calidad del software son las suites de test automatizados que testean el 100% de la funcionalidad del sistema. (¿Cómo sabe usted que cubren el 100%? Lea el siguiente artículo).

Otra métrica de calidad es cuán bien hecho está el software. El problema de la *refactorización* es actualmente un tema caliente en las conferencias de desarrollo de software. La idea es modificar el software solamente para hacerlo más simple, más legible y más mantenible, sin cambiar lo que en realidad hace. Es perfectamente posible emplear a un tercero para revisar parcelas representativas de código para determinar si cada refactorización posible fue aplicada. El beneficio de este enfoque es software órdenes de magnitud más fácil de entender y barato de cambiar: y eso es crítico para el proceso de desarrollo evolutivo que estamos proponiendo aquí. Ver [Fowler99] para mucha más información sobre la ciencia de la refactorización.

Aplicamos ambas métricas de calidad mencionadas a cada proyecto bajo el sol. Otros problemas tradicionales de calidad pueden ser mejor manejados en el contexto de las historias. La performance es uno de ellos: cualquier requisito duro de calidad debería formar parte de las historias. Un ejemplo podría ser "Todo procesamiento de final de día debe haber finalizado antes de que el banco reabra al público en la mañana siguiente". Un test correspondiente se agregaría a la batería de tests para el escenario de "proceso de final de día", y la historia sólo se consideraría entregable si el test pasa.

En otros ambientes, aspectos de calidad más livianos y transversales al sistema pueden ser importantes, como la usabilidad. En este caso, incluya algún estándar de calidad en el contrato. Lo más importante es tener una métrica indiscutible. Recuerde que la métrica no necesita ser objetiva. Por ejemplo, si a usted le preocupa la usabilidad, puede que lo mejor sea acordar acatar la decisión de un tercero, un ergonomista confiable.