Gestión clásica de proyectos vs. desarrollo ágil

Marcos Bermejo
Marc Florit
Ramon G. Sedó

PID_00212573

Universitat Oberta de Catalunya

Índice

1. Introducción

Este módulo se divide en tres partes:

1) Primero hablaremos sobre los proyectos multimedia y analizaremos su complejidad intrínseca.

2) Veremos las herramientas y procesos clásicos en la gestión de proyectos.

3) Y finalmente veremos nuevas metodologías, procesos y técnicas de gestión de proyectos multimedia que intentan corregir los problemas más habituales que surgen en el complejo desarrollo del proyecto.

2. La complejidad del proyecto multimedia

En general, los **productos multimedia** son proyectos complejos y se basan en la incertidumbre por definición. Aplicado al entorno del desarrollo de productos multimedia, podemos definir un **proyecto** como **complejo** cuando tanto la tecnología que se va a utilizar para resolver el problema como los conocimientos sobre los requisitos del mismo no son conocidos de una manera clara al principio del proyecto.

Referencia bibliográfica

Ralph Stacey. "Strategic Management and Organizational Dynamics". En: Ken Schwaber; Mike Beedle. *Agile Software Development with Scrum*.

Los requerimientos tienden a ser complejos con mucha facilidad. De hecho, podríamos hablar de requerimientos simples si:

- El cliente puede capturarlos todos y transmitirlos al desarrollador y si este es capaz de entenderlos en su totalidad.
- No existen varias partes interesadas en el proyecto con intereses divergentes.
- El cliente sabe exactamente lo que necesita.

Tal como se puede ver, esto es difícil de alcanzar. Aparte, debemos tener en cuenta que los productos multimedia los desarrollan personas que trabajan en grupo. Si las personas ya somos complejas miradas individualmente, esta complejidad se incrementa bastante cuando trabajamos en grupo.

En las metodologías tradicionales, también llamadas predictivas, se ha intentado luchar contra esta complejidad intentando hacer que tanto los requerimientos como la tecnología, ambas variables inherentes al desarrollo de productos multimedia, desaparecieran de la ecuación. Esto ha dado como resultado fases muy grandes de toma de requerimientos, largos análisis de los mismos y contratos demasiado cerrados y restrictivos.

A pesar de los esfuerzos invertidos, es difícil luchar contra la complejidad de los productos multimedia y estas aproximaciones no han llevado a la ejecución de mejores proyectos, sino a tener proyectos con problemas.

2.1. Problemas en el desarrollo de productos multimedia

Según un artículo de la IEEE (http://spectrum.ieee.org/computing/software/why-software-fails/0), los factores habituales por los que un producto multimedia falla son los siguientes:

Lectura complementaria

Podéis encontrar el artículo en http://spectrum.ieee.org/computing/software/why-software-fails/0.

- hitos del proyecto poco realistas o poco articulados,
- estimaciones poco cuidadosas de los recursos necesarios para llevar a cabo el proyecto,
- requerimientos del sistema mal definidos,
- información pobre sobre el estado del proyecto,
- riesgos no gestionados,
- comunicación pobre entre los usuarios, desarrolladores y clientes,
- utilización de tecnología poco madura,
- incapacidad de gestionar la complejidad del producto,
- prácticas de desarrollo descuidadas,
- pobre gestión del proyecto,
- presiones comerciales.

En un estudio llevado a cabo por BCS (http://www.bcs.org/content/ConWebDoc/19584), podemos ver que de los 214 proyectos estudiados de todos los ámbitos realizados con metodologías predictivas, el 23,8% fueron cancelados y el 32,2% se acabaron por encima del presupuesto o del tiempo previsto. Esto quiere decir que cerca de un 55% de los proyectos no tuvieron el final deseado. Otros estudios corroboran estos datos. Si tuviéramos en cuenta la satisfacción final del cliente con el producto entregado, seguramente estos porcentajes todavía se incrementarían más.

Referencia web

Podéis encontrar el estudio llevado a cabo por BCS en http://www.bcs.org/content/conwebdoc/19584.

Algunos de estos problemas son especialmente típicos y preocupantes:

- El cliente tarda mucho tiempo en poder utilizar el resultado del proyecto, mientras tanto, el contexto cambia y los competidores lanzan nuevos productos. Si se cancela el proyecto, el cliente habrá gastado el dinero en nada.

- El proyecto se ha complicado más de lo esperado, hay retraso y se tienen que acelerar las entregas. Empiezan los parches y no hay tiempo para pruebas ni para el control de calidad.

- El equipo hace horas extras y está poco motivado, simplemente se dedica a cumplir órdenes. Cada cual va a lo suyo.

- ¡El cliente pide cambios! Estos cambios son difíciles de implementar puesto que el proyecto está acabándose y no queda ni tiempo ni presupuesto.

- Todo el mundo acaba insatisfecho, tanto el cliente como el equipo.

2.2. ¿Por qué aparecen estos problemas?

Por un lado, tenemos que las diferentes fases se alargan más de lo previsto y siempre se ven perjudicadas las fases de pruebas y de posibles cambios.

Por otro lado, el cliente no ha visto papel (documentos de diseño) hasta una fase muy avanzada del proyecto. El cliente avanza a ciegas, gastando dinero en algo que no tiene ningún tipo de aplicación. Son documentos, no lo puede usar.

Aun así, siempre aparecen las discusiones entre cliente y proveedor. Uno cree que el otro no ha entendido lo que quería y que no se están cumpliendo los plazos previstos. El otro cree que lo que se le pide no estaba al alcance del proyecto y que está perdiendo dinero. Todo esto hace que se dedique mucho esfuerzo a objetivos que aportan poco valor.

Es decir, en el desarrollo de productos multimedia se cumple perfectamente el principio de Pareto, que afirma que el 80% del valor de un producto viene del 20% de las funcionalidades implementadas. Por lo tanto, estamos invirtiendo mucho tiempo, esfuerzo y dinero en cosas que no se usarán. ¿Tenemos que hacer pagar al cliente por eso?

Parece claro que esta no es la mejor manera de desarrollar un producto multimedia. ¿Tenemos alguna alternativa que nos pueda funcionar mejor? En esta asignatura vamos a aprender que sí.

3. Gestión clásica de proyectos

3.1. Fases de la gestión clásica de proyectos

Vista la complejidad del proyecto, la gestión clásica de un producto multimedia la dividimos en las cinco fases siguientes:

1) toma de requerimientos,
2) análisis,
3) diseño,
4) construcción,
5) pruebas.

1) Toma de requerimientos

Al inicio del proyecto el cliente sabe perfectamente lo que necesita y, además, es capaz de hacerse entender a la perfección y sabe comunicarnos sus expectativas. Además, el equipo de desarrollo sabe exactamente qué tiene que hacer para conseguir desarrollar el proyecto y se toman requerimientos detallados. Cuando esta fase acaba, ya no es necesario volver a hablar con el cliente porque ya hemos documentado, hemos especificado y hemos firmado todo lo que el cliente quiere en su producto, y hasta el final del proyecto no lo volveremos a ver, cuando le entreguemos lo que nos pidió.

2) Análisis

Una vez entra en nuestra empresa el documento (habitualmente de unas 200 páginas o más) con la especificación de los requerimientos del producto (el **qué**) recogidos en la fase de toma de requerimientos, el analista funcional lo lee y lo entiende a la perfección. Luego, basándose en este documento, es capaz de elaborar otro documento detallado de análisis funcional de estos requerimientos, junto con todo tipo de mapas y diagramas.

3) Diseño

En esta fase, una vez ha acabado la fase de análisis, el diseñador técnico es capaz de entender perfectamente el diseño funcional del que hemos hablado en la fase anterior y extrae una serie de documentos, diagramas y especificaciones técnicas para dejar por escrito todo lo que los programadores tendrán que programar (el **cómo**), para dejar claro desde el principio qué características técnicas tendrá todo el producto multimedia que estamos desarrollando.

4) Construcción

En esta fase, los programadores son capaces de tomar el diseño técnico y, sin necesidad de plantearse caminos alternativos, solo deben plasmar los documentos de la fase de diseño en código; como todo está perfectamente detallado, no aparecerá ningún error, los programadores solo tienen que seguir las indicaciones del diseño.

5) Pruebas

Finalmente, el cliente valida el producto entregado, mediante la realización de pruebas manuales que intentan reflejar el uso que se hará de la aplicación. Se elabora una estrategia de niveles de prueba, donde se practica con los diferentes componentes, módulos, y con el sistema entero sobre diferentes plataformas, entornos y contextos. El equipo de testeo (que habitualmente no es el mismo que el de construcción) lo prueba todo en el tiempo planificado. Una vez finalizadas las pruebas, en las que solo se encuentran errores menores que se corrigen inmediatamente, confirma que ha recibido lo que esperaba y no se ha de cambiar nada.

3.2. Resultados de la gestión clásica de proyectos

Durante cada una de las etapas de la gestión, en la documentación entrante se encuentra toda la información necesaria para llevar a cabo su tarea y de ahí salen otros documentos sin que en este proceso de transformación se haya perdido ninguna información por el camino. Todo este proceso fluye con total normalidad hasta que el proyecto acaba y se entrega al cliente, exactamente aquello que medio año antes había pedido.

Como hemos dicho al inicio de este módulo, esto es irreal. Nunca ha funcionado este planteamiento en un producto multimedia. ¿Por qué?

- Porque en el momento de arrancar un proyecto, cuando el cliente nos expresa sus necesidades, nunca las sabe comunicar a la perfección, además, nosotros como proveedores de producto multimedia nunca las sabemos interpretar perfectamente.

- Porque en toda esta ida y venida de documentos siempre hay errores que se van amplificando fase a fase. Un pequeño error en la toma de requerimientos se vuelve un gran error en la fase de diseño y, cuando llega a la fase de construcción, se vuelve un error incorregible con el que tenemos que convivir indefinidamente.

- Porque el cliente solo ha visto papel durante tres cuartas partes de la duración del proyecto. El cliente ha ido firmando documentación, muchas veces demasiado extensa, técnica y compleja como para entenderla y, hasta

el final, no se encuentra el resultado de todo este tiempo de espera. Y lo más probable es que no sea lo que había estado imaginando.

- Porque, siguiendo este modelo, los cambios son siempre imposibles o demasiado traumáticos. Un cambio en la fase de construcción representa muchísimo esfuerzo, es siempre caro y origen de muchos enfrentamientos con el cliente. Y la fase de construcción se vuelve un eterno "proyecto finalizado en un 97%". En este punto, los cambios de última hora o los errores arrastrados durante todo el proceso hacen que la fecha de entrega se nos eche encima y, obviamente, el producto sale sin probar y con unos índices de calidad bastante mediocres.

- Porque con esta gestión suponemos que las necesidades del cliente no variarán durante toda la duración del proyecto y eso nunca es cierto, y más en el mundo del multimedia, donde en un mes pueden cambiar muchas cosas.

4. El desarrollo ágil

En el 2001, diecisiete ingenieros de alto nivel en el campo del desarrollo de software se reunieron en Utah para explorar y compartir cuál creían que debía ser el futuro del desarrollo de software. Dentro del grupo, muchos proponían metodologías emergentes en aquel momento como Scrum, Extreme Programming, Crystal, Feature-driven development y otros. Juntos, coincidieron en el nombre para su movimiento: **Agile**.

Durante esta reunión, los integrantes constituyeron la **AgileAlliance** y escribieron el *Agile Manifesto*, un conjunto de estamentos que han servido como declaración de valores y principios que deben seguir las metodologías Agile.

- **Personas y sus interacciones** sobre procesos y herramientas.
- **Software funcionando** sobre documentación exhaustiva.
- **Colaboración con el cliente** sobre negociación de contratos.
- **Respuesta al cambio** sobre seguir una planificación.

Los valores ágiles suenan sencillos y obvios la primera vez que se escuchan y siguen sin modificarse desde el día en el que los fundadores de la Agile Alliance los publicaron por primera vez como parte del *Agile Manifesto*.

Los principios del *Agile Manifesto*

1) Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software evaluable.

2) Los cambios en los requerimientos son bienvenidos, incluido en etapas tardías del desarrollo. Los procesos ágiles aprovechan el cambio como una ventaja competitiva del cliente.

3) Liberar software con frecuencia, desde unas cuantas semanas hasta unos cuantos meses, con preferencia por la escala más corta.

4) La gente de negocio y los desarrolladores tienen que trabajar juntos a diario a lo largo del proyecto.

5) Construimos los proyectos en torno a personas motivadas. Les damos el entorno y el apoyo que necesitan, y confiamos en ellas para hacer el trabajo.

6) El método más eficiente y efectivo para compartir información con el equipo de desarrollo son las conversaciones frente a frente.

7) El software funcionando es la principal medida de progreso.

8) Los procesos ágiles promueven el desarrollo sostenible. Los patrocinadores, desarrolladores y usuarios deben ser capaces de mantener el ritmo indefinidamente.

9) La atención continua a la excelencia técnica y al buen diseño mejoran Agile.

10) La simplicidad –el arte de maximizar el trabajo no hecho– es esencial.

11) Las mejores arquitecturas, requerimientos y diseños emergen de equipos autoorganizados.

12) A intervalos regulares, el equipo reflexiona sobre cómo ser más efectivo, entonces afina y ajusta el contexto en consecuencia.

Pero llevar estos valores a la práctica en el día a día es más difícil de lo que parece inicialmente. Durante los siguientes subapartados los vamos a ir desgranando.

4.1. Las personas y sus interacciones sobre procesos y herramientas

Uno de los valores fundamentales del agilismo es que la gente que desempeña el trabajo es quien mejor sabe cómo se tiene que hacer. Al contrario, en la gestión tradicional de proyectos, en la que habitualmente existe la figura del experto o gestor que dice al resto de personas lo que tienen que hacer y cómo lo tienen que hacer, en Agile el equipo de personas que realiza el trabajo se **autoorganiza**.

Esto no implica que no se pueda sugerir al equipo que use una herramienta u otra. Una de las técnicas más utilizadas para la estimación de tareas es el **Planning Poker**, pero el equipo de desarrollo y vosotros quizás no la conoceréis nunca si no se alienta al equipo a investigar y buscar nuevas formas de trabajar, no solo a usar las que ya conocéis que funcionan. Este concepto, el de buscar la mejora continua, es necesario para lograr que el equipo de trabajo crezca y mejore, por lo tanto, cada vez es más productivo en el futuro.

4.2. Software funcionando sobre documentación exhaustiva

Este es otro de los principios ágiles que a menudo conlleva malas interpretaciones.

La documentación es adecuada cuando sirve al propósito de crear valor y mover el proyecto hacia delante, lo que permite intercambiar conocimiento.

Por ejemplo, la documentación de usuario es un recurso muy valioso en todas las herramientas de software. Los problemas se dan cuando el foco del equipo se desvía del producto que se está construyendo para centrarse en otras tareas que exigen un determinado proceso como es la documentación.

Cuando empezáis el proceso de desarrollo invirtiendo grandes esfuerzos en crear una documentación exhaustiva, sacrificáis la oportunidad de inspeccionar y adaptaros a medida que el proyecto va avanzando, aprendiendo de los errores que se van cometiendo y ajustando el proceso de desarrollo a lo que realmente necesita el proyecto.

Una mala interpretación habitual es que los equipos ágiles no documentan o planifican; pero en la práctica los equipos ágiles dedican más tiempo y energía a la planificación y documentación que los tradicionales, dado que la planificación se actualiza y se corrige continuamente.

En un desarrollo ágil de productos multimedia, la planificación y documentación se encuentra en todas partes alrededor del equipo, en forma de historias de usuario, pilas de requerimientos, tests de aceptación y gráficos grandes (¡muy grandes!) y visibles; todos ellos forman parte del entorno para una comunicación rica dentro del equipo.

4.3. Colaboración con el cliente sobre negociación contractual

Este valor ágil también busca evitar el gran diseño inicial, pero lo hace poniendo énfasis en mantener un diálogo continuo entre el equipo de desarrollo y el cliente, tan abierto y fluido como sea posible. Los contratos son buenos y necesarios, en especial cuando protegen los intereses de ambas partes, como los buenos contratos hacen.

Pero muchos de los fundadores de la Agile Alliance eran consultores y conocían los riesgos que se esconden tras los contratos. Uno de los más habituales en nuestro sector es que el proveedor se compromete a ejecutar una determinada cantidad de trabajo en un tiempo determinado, donde el provecho que saque del proyecto depende de lo rápido y barato que sea capaz de cumplir los requerimientos mínimos del cliente.

Los autores del *Agile Manifesto* concluyeron que un proyecto basado exclusivamente en un contrato pone énfasis en el lugar incorrecto. Ellos prefirieron promover un entorno colaborativo, donde el cliente y el equipo trabajan conjuntamente para alcanzar un fin que es compartido, **el éxito del proyecto**.

Ved también

El módulo "Nueva relación entre cliente y proveedor: los contratos ágiles" de esta asignatura está dedicado íntegramente a la creación de contratos ágiles, donde los explicaremos más en profundidad.

4.4. Respuesta al cambio sobre seguir una planificación

Las organizaciones que basan su modelo en seguir el plan establecido acostumbran a tener procesos creados con las mejores de las intenciones: para prevenir que el proyecto sufra retrasos, cambios de funcionalidad no planificados o cualquier otra disfunción que pueda afectar al plan inicial, no los permiten o los someten a duros requisitos, como revisar toda la planificación o incluso redactar un nuevo contrato o un anexo al actual. Pero a pesar de todo, no funciona. Un control de cambios solo funciona en un entorno donde los cambios son controlables. En cambio, un desarrollo de software es algo donde, a medida que avanzamos, vamos descubriendo algo que no sabíamos al comienzo y que nos obligará a introducir cambios no esperados.

La planificación en los proyectos de desarrollo de productos multimedia tiene que ser fluida, no fija, para el beneficio del equipo, pero sobre todo para el beneficio del producto que estamos desarrollando y, en última instancia, para el beneficio del cliente. Esta es la razón por la que planificamos para el cambio.

Bibliografía

Varios autores. *Manifesto for Agile Software Development*. Disponible en: http://agilemanifesto.org/