GAPS

Gestión Ágil de Proyectos de Software

Manuel Angel Rubio Jiménez

GAPS

Gestión Ágil de Proyectos de Software

Manuel Angel Rubio Jiménez

Resumen

Desde 2004, que comencé trabajando en el mundo de la informática dentro de varias empresas, he visto que la organización dentro de cada una de las empresas es muy similar. La creación de software, junto con su gestión se basan en unos patrones muy similares y, sobretodo, en empresas en las que el software es para ***consumo propio***, o para su explotación directa.

Este documento recoge la mayor parte de las prácticas que he usado o que he aprendido y he podido poner en práctica, junto con definiciones y conceptos sobre los entornos de desarrollo en los que me he visto envuelto. En ningún caso constituyen verdades irrefutables, pero en su mayor parte podría considerarlas buenas prácticas, al menos, ya que han ayudado a mi equipo y a mi, personalmente, a tratar con cada aspecto de la gestión del desarrollo de software a medida o del desarrollo de productos.

Historial de revisiones		
Revisión 1.0	09-05-2010	MARJ
Primer borrador.		
Revisión 1.1	11-06-2012	MARJ
Corregido y Ampliado.		

Tabla de contenidos

Una ojeada al concepto

El desarrollo de sistemas de software, de cualquier índole, tiene como problema la organización de las personas que contribuyen, de alguna forma, en la creación del proyecto de software.

Toda organización para crear un proyecto de software es diferente, según el tipo de software que se desee crear. Si el software creado es libre, para la propia organización o como producto que se pueda comercializar por parte de la empresa, esto determinará la organización propia de la propia fábrica o factoría de software.

En este documento trataré, sobretodo, la organización en caso de que la fabricación de software tenga como objetivo el uso por parte de la empresa, ya sea para uso interno (como herramienta de automatización propia del negocio) o como venta de servicios a través de la explotación de dicho software.

Capítulo 1. Tareas

La mejor estructura no garantiza los resultados ni el rendimiento, pero la estructura equivocada es una garantía de fracaso
—Peter Drucker

El primer punto en el que estoy totalmente en desacuerdo con la mayoría de factorías de software, es en controlar a las personas, las horas que trabajan en un proyecto o simplemente las horas que están sentadas en frente de una pantalla de ordenador.

Una de las primeras normas y reglas de la administración de empresas es: ***no controlar personas, sino solo los procesos***. Por lo que, el foco debe estar siempre sobre las tareas.

1. Lo más importante: las tareas

Las tareas no deberían de ser actividades aisladas, sino que deberían de formar parte de una entidad mayor como una versión de producto o un proyecto bien definido. Si el cúmulo de ideas no ligadas, una incidencia o una actividad rutinaria se desbordase, esto sería indicador ineludible de una necesidad de gestión entorno al área específica, ya que comenzaría a perder forma debido a que:

- sus responsabilidades estén muy diluidas, de modo que le llegan tareas de muchos tipos diferentes, con lo que es muy complicado agruparlas debajo de un mismo nombre;
- su forma de organizar el trabajo no conlleva el organizar dichas tareas de forma más jerárquica a través de las entidades mencionadas antes,
- el nivel de mantenimiento es tan deficiente que hace necesario un flujo constante de tareas de todos y cada uno de los proyectos ya realizados, debido a errores, adaptaciones o mejoras que no se identifican bien entre ellas ni se ajustan a un patrón bien definible.

Las tareas, para ser abarcables en el tiempo, deben de tener unas características comunes que permitan etiquetarlas, priorizarlas y agruparlas, de modo que su visión a grosso modo permitan ver si se puede postergar su realización para ajustar la estrategia de su realización a un desarrollo posterior, o su forma concreta hace que sea enlazable a otro tipo de tareas, de modo que resolviendo solo una de ellas, todas las demás quedan realizadas también.

Llegados a este punto, sería bueno dar una taxonomía interna de una tarea, es decir, su clasificación en varios niveles y por tipos:

incidencia

Son errores detectados que se notifican al equipo de desarrollo. Las incidencias a su vez pueden ser de tres tipos diferentes:

correctivo

Los errores de codificación detectados en el transcurso de la puesta en producción de un programa o descubiertos a posteriori.

adaptativo

Cuando el entorno en el que se ejecuta el programa cambia y necesita ajustes para adaptarse a los nuevos requisitos del sistema en el que tiene que ejecutarse.

perfectivo

Son los cambios solicitados para optimizar el sistema. Cuando un software se entrega, puede haber cumplido con todos los requisitos funcionales, no obstante, existen otros requisitos no funcionales, que pueden no haberse formulado o que sean implícitos y que se requieran para el correcto funcionamiento del sistema.

requisitos

Son peticiones realizadas por comerciales o marketing, siendo los primeros los enviados por parte del cliente, y los segundos la voz de lo que un conjunto de clientes, clientes potenciales y/o el mercado, los que los solicitan. Estos pueden ser de dos tipos:

funcionales

Los requisitos funcionales son aquellos que se solicitan por parte de los usuarios potenciales del sistema, o los responsables del producto en su defecto, que consisten en requisitos del funcionamiento del producto en sí, lo que puede hacer el producto y se puede probar y comprobar fácilmente.

no funcionales

Son requisitos que normalmente no piden los clientes, sino que son especificaciones que se esperan, muchas veces de forma implícita, del sistema. Se pueden, no obstante solicitar de forma explícita, pero suelen ser muy difíciles o costosos de probar (que soporte miles de usuarios, que consuma pocos recursos, qué sea rápido en realizar ciertas peticiones, etc.).

rutinas

Son las tareas que se repiten de forma recurrente, a modo de que se realicen siempre periódicamente cada cierto tiempo establecido. Este tipo de tareas suelen ser objeto de estudio para intentar su automatización.

Dentro de esta clasificación, se puede dar también el tipo de tarea ***huérfana***, aunque no la pongo en la clasificación, puesto que este tipo de tareas significará que nuestra labor de etiquetado de tareas no es lo suficientemente exhaustiva, o que quizás necesitamos refinar lo que llamamos proyecto, producto o cada una de los anteriores tipos de tareas.

Para poder controlar de forma correcta la realización de las tareas, es importante estimar el esfuerzo de las tareas y calcular la desviación de dicha tarea en base a horas, días o una puntuación que nos permita medir el esfuerzo que requiere la realización de dicha tarea.

La tarea tendrá que tener también una priorización, que corresponderá con el valor que el cliente le da a esa tarea. Es decir, si una tarea tiene un gran valor para el cliente, es susceptible de tener mayor prioridad y por tanto, mayor valor en su facturación, mientras que el tiempo que se tarde en la misma, puede ser un dato orientativo de cara al precio, más que determinante.

Nota

La imputación de horas es como una justificación de que un trabajo se ha realizado en las horas que un programador ha estado en su puesto de trabajo y, finalmente, lo que se busca es ver si todas las horas han sido provechosas, o incluso si un trabajador miente o no sobre las horas que finalmente ha imputado sobre el propio proyecto.

De cara a justificación de trabajo, no hay nada mejor que realizar el trabajo en las horas que se ha pactado que se realizará, y ante desviaciones reiteradas, convenir una forma en la que el trabajador pueda obtener mayor ayuda, ya sea en la estimación de la siguiente tarea, en su realización de forma más rápida y precisa, o incluso en su fraccionamiento, programación por parejas u otras técnicas, que permitan mayor concentración y efectividad.

2. ¿Cómo va la tarea?

El saber el estado de un proyecto a través de sus tareas, sus estados y poder conocer cosas tan importantes como los bloqueos que está teniendo un proyecto, o donde poner la presión por las dependencias que crean las tareas unas de las otras, hace que tengamos un control

mayor sobre el estado del proyecto, y podamos realizar estimaciones, sobre la marcha, para saber si el trabajo podrá concluir a tiempo.

Para que esto sea posible, cada tarea debe de tener:

estimación

La estimación puede ir determinada en base a esfuerzo o en base a tiempo, con lo que sea más cómodo trabajar al equipo. Los puntos que se indiquen reseñarán estiman la complejidad o el nivel de trabajo necesario para completar dicha tarea.

estado

Cada tarea debe de establecer un estado que indique si está en ejecución, en espera, bloqueada o terminada (hecha). Esto es importante, porque que haya una tarea bloqueada, por ejemplo, es indicativo de que se necesita o requiere actuación de más alto nivel para poder eliminar el impedimento que hace falta resolver para que esa tarea pueda seguir realizándose.

dependecia

Las tareas pueden tener lazos de dependencia con otras tareas a modo de no poder iniciarse hasta que otra tarea esté realizada o no poderse dar por finalizada hasta que la otra tarea finalice, comience o se termine. En este caso, el bloqueo estará presente en la tarea hasta que se priorice y finalice la tarea que lo bloquea.

Es importante tener presente que, la necesidad de un trabajador que está ocupado con otra tarea y no puede trabajar en la actual no indica una dependencia sobre la tarea en la que está ocupado, sino un bloqueo.

3. ¿Quién estima las tareas?

Las tareas deben de ser realizadas y estimadas por el equipo de programadores, así como por el jefe de proyecto. La estimación debe de ser parte del ejercicio común del equipo, junto con su jefe de proyecto de poner en conocimiento el trabajo a realizar, exponer las dificultades, establecer una forma de proceder previa (a grosso modo) mediante un brain storming y decidir, durante una reunión de planificación, el peso de cada una de las tareas que tiene cada uno de los requisitos, así como subdividir las tareas en caso de que sea necesario.

Las reuniones de estimación, los requisitos y las tareas son conceptos que he tomado prestados de Scrum y Kanban, por lo que se puede obtener más información revisando documentación referente a estas metodologías concretas.

4. ¿Controlamos a los trabajadores?

Sobre el control de los trabajadores, o programadores, en la mayoría de empresas se realiza haciendo que ***imputen horas***. Personalmente considero esa forma de control una desconfianza hacia los trabajadores en sí y una forma de control activa del personal que solo deriva en que las personas pierdan tiempo y generen desconfianza hacia sus superiores y compañeros. Sí, estoy en contra de los partes horarios.

El parte horario, visto desde el punto de vista funcional, solo sirve para justificar el gasto de una persona, de las horas que realiza en la empresa y en qué dice que gasta las horas en las que está en su puesto de trabajo. Esto tiene dos puntos negativos:

- el primero es que, ¿valoramos todas las horas de la misma forma?, es decir, ¿nos vale más una hora de producción máxima u ocho horas de intentar pasar lo más rápido posible escribiendo tecla a tecla una solo trozo de código sin sentido?, ¿nos fiamos de que nos ha dicho que ha gastado ocho horas en un proyecto cuando sabemos que ha entrado en páginas web como youtube, facebook y además ha desayunado largo y tendido, leído el periódico e ido tres o cuatro veces al servicio?

- el segundo, ¿reflejan realmente el trabajo que se ha estado haciendo?, ¿el gasto que se ha hecho en un proyecto determinado?

Por otro lado, hay empresas que llevan a la última expresión el control del personal a través de fichajes cada vez que entran y salen del recinto de trabajo, contabilizando el tiempo que la tarjeta ha estado insertada en la ranura de su equipo (sin la cual el equipo no funciona y la cual es necesaria para poder entrar y salir del recinto), así como un acceso excesivamente controlado a página externas como facebook, twitter, etc. En resumen, se cargan la creatividad y proactividad de sus empleados.

Capítulo 2. Requisitos y Proyectos

En todo desarrollo, hay un primer momento en el que alguien, algún comercial, alguien de marketing o nuestro jefe, se sienta con nosotros o hablando en el pasillo nos dice que habría que hacer un programa que tuviese una serie de características concretas. Eso son requisitos y el conjunto y marco en el que se engloban, un proyecto.

1. Pide, pide... Requisitos

Los requisitos son peticiones específicas, como un párrafo corto en el que el cliente, un comercial, marketing o nuestro jefe, establece una característica de un programa a realizar, mencionando una serie de parámetros o unas reglas que han de cumplirse.

Cada requisito debería de tener un forma concreta, una plantilla de característica en la que se enuncie el contexto en el que se solicita, lo que se espera que haga el programa en sí, el comportamiento que se quiere conseguir y algún ejemplo concreto que se pueda probar para explicar el caso del requisito.

Así mismo, cada requisito se puede partir después en tareas, que deben estar relacionadas o ligadas con el propio requisito. La consecución de las tareas significaría la consecución total del requisito que se puede comprobar a través del ejemplo dado en el mismo.

2. ¿Quién crea los requisitos?

Los requisitos se generan a partir de peticiones, que deben de ser validadas como requisitos. Es lógico que los comerciales, la gente de marketing o nuestros jefes nos pueden dar requisitos de forma verbal o escrita, pero ese requisito necesita algo más de trabajo para tener una forma clara (desde el punto de vista del modelo de dominio, sin ambigüedades).

Por ello, los requisitos deben de ser creados por un analista o, en su defecto, por el jefe de proyecto, tras convertir una petición en un requisito formal que sea entendible y asumible por el equipo de trabajo, así como fraccionado en tareas, que serán las unidades más pequeñas que tomará el equipo para trabajar sobre ellas.

Nota

Esto no quita que tras emitir el documento, el equipo pueda reunirse y modificar dichas tareas para subdividirlas, completarlas, mejorarlas o cambiarlas. Al final, todo es un trabajo en equipo, realmente.

3. Midiendo el trabajo por Proyectos

Para mi, un proyecto es un conjunto de requisitos que se establecen necesarios de realizar todos juntos para conformar una liberación o entrega al cliente. Cada proyecto se establece por fases bien conocidas por todos los que se hayan vistos envueltos en algún proyecto de software:

evaluación

se establece un estudio de requisitos a modo de establecer una viabilidad y valor al proyecto. El cliente en este punto tendrá que establecer su valoración de prioridad sobre todos los requisitos creados que estén sin asignar a ningun proyecto concreto.

planificación

una vez se ha realizado la evaluación de los requisitos, en esta fase se agrupan dentro del proyecto los que mayor prioridad tienen y dar una estimación de cada uno de los requisitos.

análisis

se parte cada requisito en tareas y se da la estimación de cada tarea en partición de las que se hayan dado en la fase de planificación, pudiendo ampliar o recortar en función de si la estimación previa fue menor o mayor respectivamente.

diseño

en esta fase, los implicados, se reúnen para tratar el diseño de la solución en base a infraestructura, lenguaje, librerías, etc. Los programadores, si hay más de uno implicado, pueden establecer su forma de trabajo, si se realiza de forma TDD o BDD pueden optar por un diseño emergente y terminar esta fase en cuanto se decidan los detalle de arquitectura de soporte, o si se toma la determinación de realizar un diseño previo de todo el código, especificarlo dentro de esta fase.

codificación

esta fase se puede resumir en escribir el código.

pruebas

en caso de haber empleado TDD, las pruebas se habrán escrito en la fase de codificación, con lo que se habrá combinado esta fase con la anterior de forma iterativa, siendo primero las pruebas y después la codificación. Si ha sido una codificación estándar, entonces esta sesión consistirá en desarrollar pruebas automáticas o desarrollar las pruebas de forma manual. Es bueno, no obstante, contar con un entorno de preproducción para que los encargados de infraestructura puedan montar y probar el código desarrollado a modo de poder aprender sobre su funcionamiento y comprobar que cumple todos los requisitos.

implementación

la puesta en producción.

mantenimiento

hasta este momento, las incidencias no se habían podido suceder, ya que un proyecto que se encuentra en fase de desarrollo (cualquiera de las anteriores) no puede tener fallos, en todo caso puede tener algún requisito mal desarrollado o falta de requisitos que haya que suplir con nuevas características a agregar a base de más requisitos.

finalizado

una vez el proyecto ha sido marcado como desfasado, no útil o anticuado, finaliza su mantenimiento (seguramente también porque ya no se encuentre en producción) y puede procederse a su eliminación del repositorio. Ya no recibirá más incidencias, ni tareas de ningún tipo.

4. Productos y sus dueños

Los productos ideas conceptuales sobre las que se realizan proyectos. Cada producto es un potencial desarrollo de software, el cual evolucionará agregando proyectos en su cronología de forma constante.

Cada producto está especificado por su dueño o propietario, que es quien se encarga de darle forma, especifica la mayoría de requisitos y se encarga de especificar, junto con el equipo de desarrollo y el jefe de proyecto, los requisitos que entrarán en cada proyecto.

Capítulo 3. Roles

Cada persona dentro de la empresa juega un papel importante en el desarrollo de un software. Para que el desarrollo de software se realice debe de existir un propietario que proporcione la información sobre el desarrollo a realizar, el producto, los requisitos, y estos deben de ser aceptados y formalizados por un jefe de proyecto, para que un equipo de desarrollo pueda acometerlos.

1. Propietarios de los Productos

Los propietarios o dueños de los productos son los que tienen la responsabilidad y la capacidad y conocimiento suficientes para proponer y establecer los requisitos necesarios para, a través de proyectos, ir dándole forma y evolucionar el producto de modo que pueda llegar a ser una pieza de producción y valor dentro de la empresa.

El propietario puede ser personal de marketing, comerciales, gerentes o una mezcla de los anteriores.

2. Jefes de Proyecto

Los jefes de proyecto son los encargados, dentro de un área, de acometer los proyectos derivados de los productos que caen en una o varias áreas, dependiendo de la disciplina específica que cubran.