Gestión de Riesgos

Introducción

El objetivo de todo projecto debe ser obtener un producto con la mayor calidad y minimizar el impacto de los riesgos que se pueden presentar. Aunque estas premisas son muy fáciles de comprender el jefe de proyecto necesita una forma bien fundada y práctica de poder realizar ambas cosas.

Aunque el proceso de desarrollo de software se basa en describir, planificar, asignar y revisar las tareas y recursos conocidos, existen muchos aspectos normalmente desconocidos o difíciles de establecer que pueden incidir directamente en la consecución de los resultados en el tiempo previsto y con el presupuesto establecido. Mientras que en muchos contextos los riesgos son fáciles de determinar (fallo en el envío de los proveedores, planificación inadecuada de recursos, presupuesto por debajo de lo necesario, etc.) en ingeniería de software, la propia naturaleza del producto resultante hace que los posibles riesgos a considerar sean bastante dispersos, una ubicación incorrecta de una tarea concreta puede dar lugar a retrasos significativos, la obtención y validación de los requisitos de una aplicación no depende únicamente de los analistas sino de los interlocutores, etc.

El proceso de desarrollo de software se basa fundamentalmente en aspectos conocidos. Se puede describir, planificar, asignar y revisar solo aquello de lo que se conoce como llevarlo a cabo. La gestión de riesgos incide, sin embargo, en los aspectos desconocidos.

Un riesgo siempre implica dos características:

- *Incertidumbre*(En algunos textos hablan de probabilidad). El suceso que caracteriza el riesgo puede ocurrir o no, no existen riesgos con un 100
- *Pérdida*. Si el riesgo se convierte en una realidad, ocurrirán consecuencias o pérdidas no deseadas.

En una simplificación grosera, ambos se pueden combinar en un indicador de magnitud de riesgo, estableciendo una secuencia de posibles valores en una escala ordinal: alto, significativo, moderado, menor, bajo.

Muchas empresas y organizaciones, sin embargo, trabajan en un modo de negación de riesgos: la planificación y estimación se llevan a cabo como si todas las variables fueran conocidas y suponiendo que el trabajo a realizar es mecánico (una vez dadas las condiciones previas, *!inmediatamente!* se puede comenzar la tarea siguiente) y las personas son intercambiables. En los procesos de desarrollo de software se sabe que esto no es así.

Muchas decisiones en un ciclo de vida iterativo están dirigidas por los riesgos.

Para tomar decisiones efectivas se necesita considerar los riesgos y plantear estrategias definidas que mitiguen o traten el efecto de los mismos.

Definiciones. El diccionario DRAE proporciona dos definiciones para riesgo:

- Contingencia o proximidad de un daño.
- Cada una de las contingencias que pueden ser objeto de un contrato de seguro.

También se referencia el término *correr riesgo*, como estar una cosa expuesta a perderse o no verificarse.

En el contexto de ingeniería de software, riesgo es una variable que, normalmente, puede tomar un valor que perjudica o elimina el éxito de un proyecto. De acuerdo con esto, éxito se puede considerar como la coincidencia con todos los requisitos y restricciones preestablecidos en el proyecto. El riesgo implica, casi siempre, incertidumbre o pérdida[1]. La pérdida describe el impacto que para el proyecto, que puede ser en forma de disminución de calidad del producto final, incremento de los costes, retraso de la finalización, etc. En un contexto general se habla de exposición al riesgo como el producto de la probabilidad de ocurrencia del mismo por la pérdida generada en el caso de que ocurra. Se considera éxito la coincidencia de todos los requisitos y restricciones considerados en el proyecto.

Por todo ello, se puede definir la gestión de riesgos como el análisis continuo de la situación actual para reasignar los recursos y establecer políticas de gestión contra amenazas presentes o futuras, ayudando en garantizar que ocurre lo que se había planificado inicialmente o un estado mejor. Aunque la gestión de riesgos se viene aplicando desde hace tiempo a una gran cantidad de actividades (seguros, análisis financieros, etc.) su utilización en el contexto de la ingeniería de software es relativamente reciente.

La disponibilidad de medios computacionales realmente potentes ha incidido sobre la aplicación de dichos medios de la manera mas efectiva posible. Sin embargo, se ha demostrado que no existe una relación entre el aprovechamiento corporativo de los medios y el dinero que se gasta en sistemas e ingeniería de software. A lo largo de la historia existen abundantes referencias de proyectos que no han finalizado, o que habiendo finalizado no se han utilizado nunca. Esto supone que el desarrollo de software y la gestión de riesgos están cada vez más ligados.

Categorías de riesgos

Existen diversas clasificaciones para los riesgos, según los aspectos de los mismos que se consideren.

[1] El SEI utiliza la definición de Webster Dictionary: *Riesgo es la posibilidad de sufrir pérdidas*

Considerando el aspecto de control de los mismos se pueden dividir en:

- *Directos*.- Todo aquello sobre lo que el proyecto tiene un grado de control considerable.
- *Indirectos*.- Todo aquello sobre lo que el proyecto tiene poco o ningún control.

Otra clasificación divide los riesgos en:

- *Riesgos del proyecto*, amenazan al plan del proyecto. Si aparecen puede que la planificación temporal y los costes aumenten.
- *Riesgos técnicos*, amenazan la calidad y la planificación. Identifican problemas potenciales de diseño, implementación, interfaz, etc.
- *Riesgos de negocio*, amenazan la viabilidad del sistema a construir. Los candidatos para los principales riesgos del negocio son:
  - Construir un producto o sistema excelente que, en realidad, nadie quiere (riesgo de mnercado).
  - Construir un producto que no concuerda con el interés estratégico de la empresa (riesgo estratégico).
  - Construir un producto que el departamento de ventas no sabe cómo vender.
  - Perder el apoyo de la gestión debido a un cambio de enfoque o cambios de personal (riesgo de dirección).
  - Perder presupuesto o personal asignado (riesgos de presupuesto).

Disponer de una categorización no garantiza su predicción. Otra categorización de los riesgos propuesta por Charette[2] es la siguiente:

- *Riesgos conocidos*, son aquellos que se pueden descubrir después de una cuidadosa evaluación del plan del proyecto, del entorno técnico y comercial en el que se desarrolla el mismo y otras fuentes de información fiables (fechas de entrega poco realistas, falta de especificación de los requisitos, entorno inadecuado de desarrollo, etc.).
- *Riesgos predecibles*, son aquellos que se extrapolan de la experioencia de proyectos anteriores (cambios de personal, mala comunicación con el cliente, etc.).
- *Riesgos impredecibles*, son aquellos que son muy difíciles de identificar por adelantado.

[2]Charette, R.N:, *Software Engineering Risk Analysis and Management*, McGraw-Hill, 1989

Estrategias frente al riesgo

La idea fundamental es no esperar pasivamente hasta que el riesgo se haga realidad y se convierta en un problema (se puede dar la situación de hacer fracasar el proyecto), sino decidir qué se puede hacer. Por cada riesgo percibido hay que decidir de antemano qué es lo que se va a hacer. Algunas posibles opciones son:

- Impedir el riesgo. Reorganizar el proyecto para que no pueda ser afectado por dicho riesgo.
- Transferir el riesgo. Reorganizar el proyecto de forma que algo o alguien soporte el riesgo (el usuario, el vendedor, el banco u otro elemento).
- Aceptar el riesgo. Decidir vivir con el riesgo como una contingencia. Realizar un seguimiento de los síntomas del mismo y determinar que hacer si éste se materializa.

Cuando se acepta el riesgo se deben realizar dos cosas:

- Mitigarlo. Tomar inmediatamente acciones para reducir la probabilidad o el impacto del riesgo.
- Definir un plan de contingencia. Determinar qué se debe realizar en el caso en el que el riesgo llegue a ser un problema. Dicho de otra forma, crear un "plan B".

Fuentes Bibliográficas

Tom Gilb, *Principles of Software Engineering Management*, Addison-Wesley, 1988

Elaine M. Hall, *Management Risk. Methods for software systems development*, Addison-Wesley, 1998

Shari Lawrence Pfleeger, *Software Engineering: theory and practice*, Prentice Hall, 1998

Project Management Institute, *PMBOK, Project Management Body of Knowledge*, 1996

Roger Pressman, *Ingeniería de Software: Un enfoque práctico*, quinta edición, McGraw-Hill, 2001

Walter Royce, *Software Project Management. A Unified Framework*, Addison-Wesley, 1998