Análisis de la Literatura de Gestión de Proyectos en Relación al Análisis de Riesgos en la Producción de Software

Alberto Garbajosa Poderoso

Control de la Producción Software – Ingeniería en Informática

Universidad Carlos III de Madrid

Madrid - España

100066985@alumnos.uc3m.es

Abstract: *Software projects show high failure rates due to a bad risk management. Several authors have analyzed which are the risks that appear in software engineering. This article claims to be a compilation of the ideas written by some of these authors to classify risks.*

Resumen: *Los proyectos software muestran unos elevados índices de fracaso debido a una mala gestión de riesgos. Diversos autores han analizado cuáles son los riesgos que aparecen en la ingeniería del software. Este artículo pretende ser una recopilación de las ideas escritas por algunos estos autores para clasificar los riesgos.*

Keywords: *Producción software, gestión de riesgos, causas del fracaso*

1. Introducción

El desarrollo de software es un proceso complejo que involucra diferentes fases por las que es necesario pasar hasta llegar al producto final entregado al cliente.

Es común que en alguna de estas fases se produzcan errores que modifiquen la planificación establecida y ralenticen la marcha del proyecto. Estos errores, además, suelen llevar asociado un sobrecoste económico que las empresas, en ocasiones, son incapaces de asumir.

De este modo, se sabe que existen unos elevados porcentajes de fracaso de los proyectos software debido a errores para los que no se había previsto una solución adecuada.

Es por ello que existe gran interés en establecer planes de gestión de riesgos adecuados para cada proyecto, que prevean posibles riesgos, garantizando una respuesta eficiente en caso de que llegaran a hacerse efectivos.

Para ser capaces de elaborar estos planes es importante conocer cuáles son los riesgos más comunes que se presentan durante el proceso de desarrollo de un producto software. Conocerlos y clasificarlos puede ayudar a entender con más claridad cuál es la mejor manera de abordarlos.

Con este artículo se pretende hacer una recopilación de las ideas que diversos autores han mostrado para llevar a cabo este entendimiento y clasificación de los riesgos.

2. Antecedentes.

Antes de empezar a clasificar los riesgos, es importante tener claro qué se entiende por "riesgo". Según el diccionario de la RAE, riesgo es: *"Contingencia o proximidad de un daño*". Esto quiere decir que un riesgo es un problema potencial, algo que aún no ha ocurrido pero que puede suceder, con consecuencias negativas.

Un riesgo tiene dos características que lo definen y que han de ser tenidas en cuenta:

- La probabilidad de que el riesgo se haga efectivo.
- El coste asociado.

Conociendo estas características, se puede tener un mayor control de cada riesgo, lo que repercute en la manera en la que se establece el plan para evitarlos o controlarlos.

Sin embargo, antes de llegar a este nivel de detalle sobre cada riesgo en particular, es necesario tener una visión más amplia sobre los tipos de riesgos que pueden aparecer en el proyecto. Es importante clasificarlos, puesto que de esa manera se podrá dar una respuesta rápida (y efectiva) a riesgos del mismo tipo, para los que exista un orden de prioridades o un plan de solución.

En el siguiente apartado se comentarán varias maneras de clasificar los riesgos en la producción software.

3. Formas de clasificar los riesgos.

Una primera clasificación [1] se basa en la idea de que los proyectos que se realizan son diferentes entre sí y, por tanto, los riesgos a los que se enfrentan no les afectan de la misma manera.

Así, en el artículo los autores dividen los proyectos en tres categorías, proyectos de bajo riesgo, de riesgo medio y de alto riesgo.

Para todos ellos evalúan, de entre todas las características de un proyecto, las tres siguientes:

- Duración del proyecto: cuanto más largo es, más aspectos hay que tener en cuenta y se asume que el riesgo es mayor.
- Subcontratación: es más arriesgado contratar a empresas externas para que participen en determinados aspectos del proyecto que directamente controlar personalmente la producción dentro de la propia empresa.
- Orientación estratégica del proyecto: distinguen proyectos estratégicos (para mejorar determinados procesos, proporcionar ventajas competitivas a la empresa) de proyectos transaccionales (captura y procesamiento de datos) o de proyectos informativos (proporcionar información, ayuda a la toma de decisiones).
- La razón de escoger esas tres era que para ninguna de ellas se había demostrado empíricamente su influencia sobre el riesgo con anterioridad.

Tras realizar el estudio, pudieron demostrar la influencia de los factores sobre los riesgos en la producción software y la conclusión a la que llegaron es que:

- A los proyectos de bajo riesgo les afecta la “complejidad”: de tareas, del uso de la tecnología, el tamaño y duración del proyecto...
- A los proyectos de alto riesgo, les afectan en mayor medida todos los riesgos relacionados con el tema de requisitos, planificación y control: estimaciones y planificación inadecuadas, falta de liderazgo, poca o mala comunicación con el resto del equipo...

Otros autores han estudiado la influencia que tienen los riesgos sobre el éxito y la eficiencia en un proyecto software [2].

Su estudio revela que los riesgos de que se produzcan errores al tratar con los datos se controlan, en general, de manera adecuada. Sin embargo, existen otros riesgos que afectan de forma crítica al proyecto y que están relacionados con el equipo de trabajo. Se pueden resumir en:

- Falta de experiencia de los miembros del equipo, que no saben trabajar con objetivos poco claros, sin una dirección precisa, con poca capacidad de trabajo en equipo o con falta de adaptación a nuevos sistemas. Se recomienda que, si se diera el caso de contar con equipos así, se trataran de mejorar las habilidades de los miembros (mediante cursos, por ejemplo) antes de comenzar el proyecto.
- Definición imprecisa de los roles de los miembros del equipo, no sabiendo cada persona hasta dónde puede llegar y entremezclándose las competencias. Asimismo, la comunicación entre los miembros del equipo empeora, al carecer de un referente claro al que consultar en caso de conflicto. De nuevo se recomienda, antes de empezar el proyecto, reunirse con los miembros del equipo y establecer de forma clara y precisa las competencias de cada uno para evitar problemas una vez que se ha empezado a trabajar.

De forma intuitiva, se ve cómo ambos afectan a la efectividad del desarrollo del proyecto, puesto que si el equipo no está bien estructurado, y la comunicación entre miembros es deficiente, se perderá mucho tiempo en realizar tareas que deberían ser más sencillas de otro modo, haciendo que haya tareas que no puedan completarse o bien ralentizando la marcha del proyecto (y por tanto encareciendo el resultado).

Otro de los estudios analizados [3] clasifica los riesgos en cuatro cuadrantes (Ilustración 1) en base a su importancia y al nivel de control que requieren.

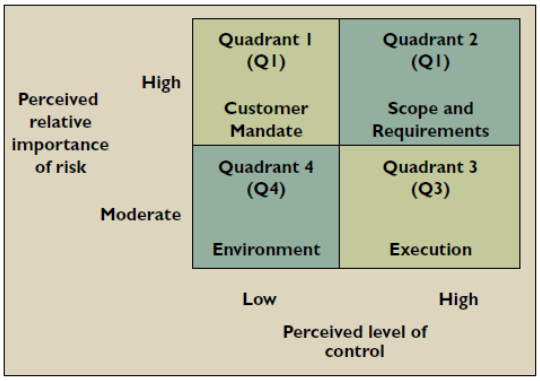

Ilustración 1: Cuadro de clasificación de riesgos [3].

El primer cuadrante (Q1) se centra en los riesgos relacionados con el cliente o los usuarios finales del sistema, como puede ser la falta de participación del cliente que no se involucra en el desarrollo del sistema, lo cual acaba perjudicando a la correcta marcha de éste.

El segundo cuadrante (Q2) se basa en el alcance y los requisitos del sistema. Por ejemplo, el contar con un jefe de proyecto que no sea capaz de evaluar de forma previa al desarrollo cuál va a ser el alcance del sistema, supone un riesgo para el proyecto.

El tercer cuadrante (Q3) se centra en el desarrollo. Riesgos pertenecientes a este cuadrante son, por ejemplo, el no contar con una metodología de desarrollo adecuada, el definir de forma poco clara los roles del equipo o el hecho de elaborar una mala planificación.

El cuarto y último cuadrante (Q4) trata del entorno de trabajo, tanto interno como externo. Por ejemplo, cambios en el personal del equipo de desarrollo o cambios de dirección del proyecto suponen riesgos que es necesario saber afrontar.

La conclusión a la que llega ese estudio es que de las cuatro divisiones de los riesgos, hay algunas más importantes que otras.

Así, Q4 (entorno) se considera el cuadrante de menor importancia, mientras que Q2 (alcance) y Q3 (ejecución) son críticos para el éxito en un proyecto de desarrollo software, puesto que se centran en el proceso y no en aspectos externos.

También advierten de que según cuál sea el enfoque del proyecto (alcance, coste o tiempo) habrá que manejar los riesgos de diferente manera centrándose más en los apartados requeridos.

En resumen, la recomendación que hacen los autores es siempre tratar de minimizar los riesgos relacionados con el proceso (Q3) y, en la medida de lo posible, hacer lo mismo con los relacionados con el producto (Q1 y Q2) de acuerdo a las prioridades internas y a la forma que se tenga de organizar el proyecto.

El siguiente estudio analizado [4] muestra algunos principios a tener en cuenta al desarrollar software para evitar caer en riesgos innecesarios que se acabarán lamentando a medida que avanza el proyecto.

Por ejemplo, un problema común al emplear el ciclo de vida en cascada es que se comienza a definir demasiado pronto una serie de requisitos sin haber llegado aún a comprender las implicaciones (y los riesgos) asociados a ellos.

Pero la solución no pasa por usar otro ciclo de vida más dinámico (desarrollo evolutivo) ya que surgen nuevos riesgos que antes no aparecían. Por ejemplo, es común según el artículo que una persona tenga una idea de mejora para el sistema y decida añadirla por su cuenta, pensando en ajustar posteriormente el sistema en el caso de que no permita adaptar esa nueva característica. Sin embargo, aunque esto pueda funcionar en algunos dominios, para proyectos complejos resulta arriesgado, por añadir nuevas implicaciones que hay que tener en cuenta y que hacen aparecer nuevos riesgos.

El artículo identifica los diez riesgos más importantes en el desarrollo de software:

1. Carencias del personal.
2. Calendario de trabajo y presupuesto inadecuados.
3. Desarrollo incorrecto de funciones y propiedades.
4. Desarrollo incorrecto de la interfaz de usuario.
5. "Gold-plating": la realización de tareas adicionales e innecesarias.
6. Flujo continuo de cambios en los requisitos.
7. Deficiencias en componentes realizados externamente.
8. Deficiencias en tareas realizadas externamente.
9. Deficiencias en el rendimiento a tiempo real.
10. Forzar las capacidades de la informática.

En la Ilustración 2 se puede ver cada uno de estos riesgos junto con las técnicas que los autores recomiendan para solucionarlos.

TOP 10 SOFTWARE RISK ITEMS.

Risk item	Risk-management technique
Personnel shortfalls	Staffing with top talent, job matching, team building, key personnel agreements, cross training.
Unrealistic schedules and budgets	Detailed multisource cost and schedule estimation, design to cost, incremental development, software reuse, requirements scrubbing.
Developing the wrong functions and properties	Organization analysis, mission analysis, operations-concept formulation, user surveys and user participation, prototyping, early users' manuals, off-nominal performance analysis, quality-factor analysis.
Developing the wrong user interface	Prototyping, scenarios, task analysis, user participation.
Gold-plating	Requirements scrubbing, prototyping, cost-benefit analysis, designing to cost.
Continuing stream of requirements changes	High change threshold, information hiding, incremental development (deferring changes to later increments).
Shortfalls in externally furnished components	Benchmarking, inspections, reference checking, compatibility analysis.
Shortfalls in externally performed tasks	Reference checking, preaward audits, award-fee contracts, competitive design or prototyping, team-building.
Real-time performance shortfalls	Simulation, benchmarking, modeling, prototyping, instrumentation, tuning.
Straining computer-science capabilities	Technical analysis, cost-benefit analysis, prototyping, reference checking.

Ilustración 2: Tabla de riesgos y solución [4].

Posteriormente establece una relación entre una situación del proyecto y la probabilidad de que surja el riesgo afectando a los costes. Por ejemplo, un proyecto con gran estabilidad en los requisitos, es muy improbable que vea afectados los costes iniciales.

De este modo, con el uso de la tabla siguiente (Ilustración 3) se puede obtener una ayuda sobre la probabilidad de cada uno de los factores, y uniéndolas se tendrá así una idea de cuál es el riesgo de que el coste estimado para el proyecto se vea afectado.

QUANTIFICATION OF PROBABILITY AND IMPACT FOR COST FAILURE.

Cost drivers	Probability Improbable (0.0-0.3)	Probable (0.4-0.6)	Frequent (0.7-1.0)
Requirements			
Size	Small, noncomplex, or easily decomposed	Medium to moderate complexity, decomposable	Large, highly complex, or not decomposable
Resource constraints	Little or no hardware-imposed constraints	Some hardware-imposed constraints	Significant hardware-imposed constraints
Application	Nonreal-time, little system interdependency	Embedded, some system interdependencies	Real-time, embedded, strong interdependency
Technology	Mature, existent, in-house experience	Existent, some in-house experience	New or new application, little experience
Requirements stability	Little or no change to established baseline	Some change in baseline expected	Rapidly changing, or no baseline
Personnel			
Availability	In place, little turnover expected	Available, some turnover expected	Not available, high turnover expected
Mix	Good mix of software disciplines	Some disciplines inappropriately represented	Some disciplines not represented
Experience	High experience ratio	Average experience ratio	Low experience ratio
Management environment	Strong personnel management approach	Good personnel management approach	Weak personnel management approach
Reusable software			
Availability	Compatible with need dates	Delivery dates in question	Incompatible with need dates
Modifications	Little or no change	Some change	Extensive changes
Language	Compatible with system and maintenance requirements	Partial compatibility with requirements	Incompatible with system or maintenance requirements
Rights	Compatible with maintenance and competition requirements	Partial compatibility with maintenance, some competition	Incompatible with maintenance concept, noncompetitive
Certification	Verified performance, application compatible	Some application-compatible test data available	Unverified, little test data available
Tools and environment			
Facilities	Little or no modification	Some modifications, existent	Major modifications, nonexistent
Availability	In place, meets need dates	Some compatibility with need dates	Nonexistent, does not meet need dates
Rights	Compatible with maintenance and development plans	Partial compatibility with maintenance and development plans	Incompatible with maintenance and development plans
Configuration management	Fully controlled	Some controls	No controls
	Impact		
	Sufficient financial resources	Some shortage of financial resources, possible overrun	Significant financial shortages, budget overrun likely

Ilustración 3: Tabla de probabilidad e impacto sobre el coste [4].

El artículo comenta que la priorización de riesgos puede hacerse con ayuda de la tabla, ya que permite auto-evaluar los riesgos en función del proyecto concreto que se vaya a realizar.

Tras comentar algunos aspectos sobre cómo gestionar los riesgos (que no se incluyen en este artículo), el autor concluye dando una idea importante: que el análisis de riesgos no es un libro de recetas, sino que se requiere sentido común para ser capaces de evaluar adecuadamente y con antelación los posibles problemas que se plantearán durante el ciclo de vida del proyecto.

El último de los artículos analizados [5] clasifica los riesgos de un proyecto software en siete categorías:

1. Relaciones comerciales y legales.

    a) Contratación de terceros inadecuada, que no se ajustan al propósito del proyecto.
    b) Falta de protección de los derechos de autor, que puede hacer que otros se aprovechen del trabajo realizado.
    c) Disputas entre cliente y empresa, con problemas para llegar a acuerdos.

2. Circunstancias económicas.

    a) Cambios en las condiciones del mercado.
    b) Alta competitividad, agresiva, que es capaz de hacer lo mismo de forma más eficiente (más rápido o más barato).
    c) Software que ya no se va a necesitar, y provoca que se termine el proyecto de forma prematura, porque su valor disminuye y hace que sea más caro seguir gestionándolo.

3. Comportamiento humano.

    a) Carencias del personal, incapaces de cumplir las tareas que se le asignan.
    b) Carencias de dirección, con poca experiencia y capacidad de motivación.

4. Circunstancias políticas.

    a) Poco apoyo corporativo, cuando la empresa da prioridad a su propia agenda antes que a realizar un seguimiento del proyecto, o cuando aparecen cambios de dirección, de organización interna, etc.
    b) Poco apoyo ejecutivo, cuando los objetivos del proyecto no se cumplen debido a problemas entre departamentos dentro de la propia empresa o con empresas externas.
    c) Colección de requisitos no relacionados con el proyecto, añadidos por motivos políticos, que ralentizan el proyecto.

5. Tecnología y problemas técnicos.

    a) Documentación de usuario pobre o inadecuada.
    b) Software que no se ajusta fielmente a su propósito.
    c) Rendimiento pobre en la producción del sistema.
    d) Limitaciones técnicas para llegar a una solución.
    e) Requisitos incompletos.
    f) Interfaz de usuario inapropiada para el cliente.

6. Controles y actividades de dirección.

    a) Calendario de trabajo y presupuesto inadecuados.
    b) Continuos cambios en los requisitos por parte del cliente.
    c) Falta de aceptación total por parte del cliente, que alarga el cierre del proyecto.

d) Problemas para verificar el trabajo diario, lo que repercute negativamente en la realización de posibles re-planificaciones, que serían necesarias para corregir las desviaciones del proyecto.
e) Falta de un único punto de responsabilidad, que sirva como referencia, ya que en ocasiones es común que existan varios líderes dentro del equipo, pero no se sabe claramente quién es el responsable.
f) Falta de liderazgo.
g) Desarrollo de funcionalidades incorrectas.
h) Falta de procesos formales de gestión del cambio.

7. Actividades individuales.

a) Centrarse en detallar más de lo debido determinados aspectos del sistema, descuidando objetivos primordiales.
b) Expectativas irreales de entrega, optimistas tanto en coste como en tiempo.

A partir de esta clasificación, en el artículo se trata de demostrar la importancia de cada riesgo identificado. Así, se comenta que el 60% de los riesgos tienen consecuencias graves (23% muy probables, 19% medio y 18% poco probables), lo que demuestra que embarcarse en un proyecto software es una tarea arriesgada, justificando así las elevadas tasas de fracaso que presentan en el sector.

En cuanto a las principales causas del riesgo se detalla que son:

- Carencias del personal.
- Calendario de trabajo y presupuesto inadecuados.
- Expectativas optimistas.
- Requisitos incompletos.
- Disminución de oportunidades debido a entregas tardías del software.

Es decir, que de todos los riesgos que pueden surgir en un proyecto, los que suponen mayor inconveniente no tienen que ver con problemas técnicos, sino con la manera en que se lleva a cabo la dirección del proyecto.

Por ello se recomienda tener siempre muy en cuenta la gestión de los aspectos relacionados con el alcance y la calidad del sistema, la gestión de los recursos humanos y materiales con los que se cuenta y la gestión de los clientes (*stakeholders*). Además, se recomienda a los jefes de proyecto mantenerse alejados de los problemas técnicos, para así poder ocupar su tiempo en esas tareas de gestión.

4. Conclusiones

La conclusión principal que se puede extraer de este artículo es que no existe una única manera de clasificar los riesgos que pueden aparecer en la producción de software, a pesar de que las propuestas explicadas puedan contener ciertas similitudes.

Por ello, puede darse el caso de que, evaluando un proyecto que se va a realizar desde una perspectiva, se llegue a la conclusión de que existe “poco riesgo”; mientras que, si se evalúa con otros criterios, se termine por

concluir que es un proyecto arriesgado que requiere más atención (y por tanto mayor inversión para evitar los problemas que puedan surgir).

Entonces, ¿cuál es la manera correcta de evaluar los riesgos de un proyecto?

Hay que tener en cuenta que la Ingeniería del Software es relativamente joven, sobre todo si se compara con otras disciplinas que han ido evolucionando para encontrar maneras más eficientes de cumplir su cometido desde hace siglos. Aún nos encontramos en un punto en el que la tasa de fracaso en el desarrollo de proyectos software sigue siendo muy elevada.

Dicho esto, la mejor opción para asegurar el éxito pasa por el conocimiento. Los responsables del proyecto tienen que conocer al cliente, el personal del que disponen y el tiempo y presupuesto con el que cuentan. Además han de tener muy claro qué es lo que quieren conseguir con el proyecto, cuál va a ser el alcance del sistema que se va a desarrollar y qué es exactamente lo que el cliente espera encontrar al finalizar el ciclo de vida.

Y así, basándose en su experiencia y en su propio criterio (no hay que olvidar la importancia del "sentido común" mencionado en [4]), los responsables del proyecto tienen que ser los que evalúen cada uno de manera individual, examinando de todos los posibles riesgos conocidos, aquéllos que consideren los factores más importantes para cada caso, y que tendrán que controlar en mayor medida.

Finalmente, y en relación con esto mismo, se considera que es recomendable también documentar siempre el plan de riesgo de cada proyecto realizado, tanto los riesgos identificados como la manera en la que se han intentado evitar o resolver. De este modo, independientemente del resultado del proyecto, esa información será de vital importancia para proyectos futuros, permitiendo adquirir esa experiencia requerida para lograr conseguir que aumente la tasa de éxito en la producción de software.

Referencias

[1] Wallace, L. & Keil, M. & Rai, A. (2004). *Understanding project risk: a cluster analysis.* Information and management 42 (2004) 115-125.

[2] Jiang, J. & Klein, G. (1999). *Software development risks to project effectiveness.* The Journal of Systems and Software 52 (2000) 3-10.

[3] Wallace, L. & Keil, M. (2004). *Software project risks and their effect on outcomes.* Communications of the ACM 47 (2004).

[4] Boehm, B. W. (1991). *Software Risk Management Principles and Practices.* Defense Advanced Research Projects Agency.

[5] Baccarini, D. & Salm, G & Love, Peter E. D. (2004). Management of risks in information technology projects.

[6] Industrial Management & Data Systems 104 (2004) 286-295.