

Le risk management : peut-on s'en passer ?

Comment faire en sorte d'anticiper les problèmes ? C'est l'art, entre autres, du chef de projet.

Au début d'un projet, le responsable de projet s'appuie sur son expérience pour identifier les éléments susceptibles de poser problème par la suite. Il dispose pour ce faire d'une liste plus ou moins fournie d'événements qui, lors de projets précédents, ne se sont pas passés comme prévu (comprenez qui ont engendré un retard ou un surcoût). C'est sur cette base qu'il peut identifier les risques.

Gérer les risques consiste à :

- identifier tous les points faibles du projet
- faire en sorte que les problèmes qui potentiellement en découleraient aient un impact réduit sur le déroulement du projet.

Pourquoi la gestion des risques ?

La résolution d'un problème coûte beaucoup plus cher en fin de projet qu'au début. Un exemple caricatural est celui d'une architecture qui ne supporte pas la charge, suite à un défaut de conception.

Si l'on s'en rend compte une fois l'application terminée, la résolution du problème peut impacter des milliers de lignes de code et sera donc onéreuse. En revanche, si le problème est détecté au début de projet, une refonte de l'architecture aura nécessairement un impact restreint. La gestion des risques a donc pour objectif principal de contribuer à maîtriser le

coût d'un projet. Le chef de projet est responsable de la gestion des risques. Le développeur n'est en général pas concerné par cet aspect d'un projet, si ce n'est indirectement par le fait que le chef de projet va tenir compte des risques pour affecter les tâches aux membres de l'équipe. Le rôle du directeur de projet, lorsque ce rôle est prévu dans l'organisation, est de s'assurer que le chef de projet prend en considération des risques raisonnables et met en place les actions adéquates.

Comment gérer les risques ?

La première étape est d'identifier les risques à partir des informations disponibles sur le projet. Le chef de projet s'appuie sur son expérience et sur l'avis de l'architecte technique et de l'analyste fonctionnel. Lorsque les risques ont été identifiés, ils sont priorisés. Pour chaque risque, le chef de projet évalue sa probabilité d'occurrence et le surcoût engendré (ou l'impact sur l'atteinte des objectifs stratégiques du projet) dans l'éventualité où le risque deviendrait avéré. Le chef de projet définit un seuil d'alerte (lorsque cela a un sens), permettant de mesurer objectivement si le risque est avéré. Par exemple, le seuil d'alerte peut être un nombre de modifications de spécifications sur une période donnée, ou bien un temps de réponse supérieur à une certaine va-

leur. Les risques peu probables et ayant peu d'impact sur le projet seront juste surveillés. Les risques très probables et pouvant avoir un impact important seront traités par des actions préventives. Un exemple de risque important est l'utilisation, sur le projet, de technologies (bibliothèque, logiciel, matériel,...) sur lesquelles les membres de l'équipe ne sont pas expérimentés. Ce risque sera traité, en général, par des actions préventives de formation et de sollicitation d'experts. Pendant tout le déroulement du projet, les risques seront régulièrement suivis par le chef de projet qui tient à jour leur évaluation et supervise les actions lancées dans ce cadre. Le projet a d'autant plus de chances d'atteindre ses objectifs que l'importance des risques est rapidement diminuée.

Avec quels outils ?

Un projet typique (quelques hommes-mois) comporte une dizaine de risques. Si le chef de projet identifie plusieurs dizaines de risques sur un projet, c'est qu'il est impératif de remanier considérablement ses objectifs. Un simple tableau est donc en pratique suffisant pour la gestion des risques, il n'est pas utile de prévoir la mise en œuvre d'un logiciel spécifique. Les écarts de compréhension entre les utilisateurs de l'application et les développeurs sont fréquents. De plus, les utilisateurs changent

Identifiant	Description du problème	Causes	Probabilité	Impact	Critéris	Seuil d'alerte	Actions
R01	Non adhésion des filiales	A priori négatif (résistance au changement, enjeux politiques) Mauvaise appréhension du besoin	Moyen	Fort	Critique		Prendre en compte leurs besoins Présenter en amont le produit envisagé Marketing Design
R02	Non maîtrise du processus	Manque d'expérience dans LP	Fort	Fort	Critique		Formation Coaching
R03	Problèmes de communication et concertation	Eloignement géographique des acteurs	Fort	Fort	Critique	Retours importants sur des artefacts	Prévoir des transmissions fréquentes des artefacts Déplacements réguliers pour des réunions, à défaut visioconférence ou conférences téléphoniques
R04	Disponibilité des acteurs	Indisponibilité des acteurs Mêler en raison de réorganisations Acteurs à temps partiel	Fort	Fort	Critique	Avancement des spécifications non conforme à l'avancement planifié	Alerter la direction pour donner des directives permettant de rendre les acteurs disponibles Ressources supplémentaires pour permettre aux acteurs clés de décharger du temps pour le projet
R05	Délaï trop court	Période minimal trop important	Moyen	Fort	Critique	Délaï très différent du délaï déterminé à partir de la charge estimée	Estimer au plus tôt le charge minimale nécessaire Demander un arbitrage pour ne réaliser qu'un premier sous-ensemble
R06	Solution non maintenable	Défaüt de conception	Moyen	Moyen	Majeur	Non-conformité aux normes de développement Coût annoncé des modification important	Utilisation de la technologie objet Normes de réalisation à définir Formation des développeurs Revue de conception régulières Revue de code régulières
R07	Solution non réutilisable	Prise en compte du besoin de la filiale, coût d'adaptation à d'autres filiales important	Moyen	Faible	Mineur		Impliquer des utilisateurs autres que ceux de la filiale
R08	Objectifs de performances non atteints	Architecture inadaptée	Faible	Fort	Majeur	Temps de réponse constatés importants	Prototyper l'architecture Mesurer régulièrement les performances Contrats de service avec les fournisseurs (matériel, réseau)
R09	Non respect des objectifs de disponibilité	Architecture inadaptée	Faible	Moyen	Majeur		Tester les solutions de clustering/backup dès que possible
R10	Dépassement des coûts et délais		Fort	Fort	Critique	Avancement de la réalisation non conforme aux prévisions	Suivi de l'avancement Mise en œuvre du processus unifié
R11	Problèmes techniques	Mise en œuvre d'une nouvelle technologie	Faible	Moyen	Majeur		Formation Intervention d'experts Prototypage

Exemple d'un tableau de bord de gestion de risques, appliqué à un projet.

parfois d'avis après avoir réfléchi, ou suite à un échange avec d'autres personnes par exemple. Le risque associé correspond au problème potentiel de la livraison d'une application qui ne répond pas correctement aux besoins des utilisateurs. La principale action préventive pour ce risque consiste à utiliser une démarche itérative (par exemple le processus unifié), par petites étapes, qui permet de réduire le fossé de compréhension entre les différents intervenants du projet. L'architecture technique a pour vocation de répondre à un certain nombre de besoins considérés comme non

fonctionnels : disponibilité de l'application, tenue en charge, temps de réponse, sauvegardes, sécurité, intégrité des données, ... Il importe de s'assurer au plus tôt que les choix effectués permettent effectivement de répondre au cahier des charges. Un exemple classique est le test de performances exécuté a posteriori. Si l'application répond aux exigences de temps de réponse du premier coup, c'est parfait, mais malheureusement rare. Dans le cas contraire, les optimisations s'avèrent souvent onéreuses et longues. De plus, les modifications de l'application effectuées à cette occasion vont nécessiter une nouvelle passe de tests de non régression.

Un cas fréquent de dépassement budgétaire est lié à l'intégration dans l'équipe projet de per-



La gestion du risque passe aussi par la sécurité dans sa politique. Ici, l'outil SmartTrace

sonnes n'ayant aucune connaissance d'une technologie qui pourtant s'avère fondamentale pour le projet. La stratégie souvent mise en oeuvre consiste à laisser la personne s'auto-former sur le tas, ce qui conduit à un code de piètre qualité, réalisé avec beaucoup d'efforts, et qui fonctionne mal.

Il est donc important sur un projet d'anticiper ce risque par des actions de formation et des contrôles réguliers du code par un expert (l'architecte en général). La mise en oeuvre de deux logiciels sur un même serveur est potentiellement risquée. Il est arrivé par exemple qu'un défaut dans l'utilisation de la mémoire partagée entre plusieurs programmes limitait la mémoire du processus principal de l'application à 512 Mo, alors que la machine avait 4 Go de mémoi-

re, déterminée en fonction du nombre d'utilisateurs. Ce type de risques est traité par la réalisation d'un prototype, souvent jetable, dont l'objectif est de s'assurer que les choix effectués fonctionnent effectivement en pratique.

Conclusion

La gestion des risques dans un projet informatique, sauf cas très particulier, est absolument indispensable pour la maîtrise des coûts et du délai. Dans tout projet de taille industrielle, des imprévus surviennent inévitablement.

La gestion des risques consiste à anticiper les problèmes et à faire en sorte que leur impact sur le projet soit limité. Ne pas gérer les risques, c'est donc avec quasi-certitude s'exposer à des retards et surcoûts préjudiciables à la fois au maître d'oeuvre mais aussi au maître d'ouvrage.

■ Dominique Méra

Consultant senior d'Objet Direct (filiale de Homsys Group)

A propos de Objet Direct et Homsys Group :

Créé en 1991, Homsys Group est spécialisé autour de la Business Intelligence (Homsys) et les technologies Objet et Internet (Objet Direct). Homsys Group est implanté à Paris, Marseille, Lyon, Toulouse, Bordeaux, Grenoble et Rennes.

www.objetdirect.com / www.homsysgroup.com

Gestion du risque en logiciel libre

AVIS D'EXPERT



D'aucuns affirment que la migration vers un environnement open source (ou "logiciel libre") est gage d'économie et de pérennité. C'est en partie vrai, mais en dépit de l'agitation actuelle autour du concept du logiciel libre, il convient d'identifier les risques d'une telle migration. Il y a deux ans, un grand groupe industriel français nous confie la conduite d'un projet sensible. Cette division commercialise

des systèmes de gestion de parkings publics. Des systèmes sont installés un peu partout dans le monde, dans des centres-villes ou des aéroports. La technologie est vieillissante (le logiciel est fiable mais tourne sous MS-DOS) et les clients font pression pour que le produit intègre les technologies natives comme le support du réseau Ethernet/TCP-IP. Les contraintes sont fortes :

- Le matériel actuel (prévu pour MS-DOS) doit être conservé.
- La fiabilité de la nouvelle solution doit être parfaite et l'interface utilisateur ne doit pas être modifiée (utilisation par le public).
- L'interface de programmation des applications ne doit pas être modifiée (utilisation de sous-traitance).
- Le code source C++ de l'application doit être modifié le moins possible et rester indépendant du système.

Nous déléguons un expert chargé d'évaluer la faisabilité du projet. Les principes sont simples : discuter avec les développeurs, analyser quelques portions de codes sensibles et en déduire les composants libres utilisables.

De cette phase nous déduisons une nouvelle architecture, un coût et un délai, sans oublier la partie légale, le client ne doit surtout pas être en porte à faux vis-à-vis des licences libres, nul n'est censé ignorer la GPL ! (General Public License)

Nous choisissons des développeurs expérimentés capables d'effectuer un minimum de transfert de compétences vers les équipes client. Coder n'est pas suffisant, il faut permettre à l'équipe client de s'approprier le nouveau code source en douceur, sans devenir des spécialistes. L'indépendance du code avec Linux est fondamentale vu les garanties de maintenance dans le temps que le client assure sur le logiciel.

Nous utilisons fréquemment des tests de non régression, l'antique compilateur MS-DOS Borland C++ n'a rien d'un GNU C++ dernière mode. Un défaut de compilation caché serait dramatique en exploitation publique.

■ Pierre Ficheux

CTO Open Wide - pierre.ficheux@openwide.fr