Assurances

Assurances

Quelques réflexions sur l'informatique

André Parizeau

Volume 55, Number 3, 1987

URI: https://id.erudit.org/iderudit/1104585ar
DOI: https://doi.org/10.7202/1104585ar

See table of contents

Publisher(s)

HEC Montréal

ISSN

0004-6027 (print)
2817-3465 (digital)

Explore this journal

Cite this document

Parizeau, A. (1987). Quelques réflexions sur l'informatique. *Assurances*, *55*(3), 408–415. https://doi.org/10.7202/1104585ar

Article abstract

In his article, Mr. André Parizeau expounds on the elements of a well-managed project: a clearly defined commitment from all parties concerned, the proper planning, from the outset, of all its phases, a judicious analysis of the choices to be made throughout the project and, finally, effective communication between the various groups involved.

Quelques réflexions sur l'informatique

par

M. André Parizeau[1]

In his article, Mr. André Parizeau expounds on the elements of a well-managed project : a clearly defined commitment from all parties concerned, the proper planning, from the outset, of all its phases, a judicious analysis of the choices to be made throughout the project and, finally, effective communication between the various groups involved.

~

Pour plusieurs cadres d'entreprises, l'informatisation de leurs opérations est généralement vue comme une nécessité. Parallèlement, l'informatique est souvent comprise, à la fois comme *la solution* à tous les problèmes et comme une source constante de maux de tête.

Que l'informatique soit devenue une nécessité pour progresser dans quelque secteur que ce soit est aujourd'hui une évidence sur laquelle il serait inutile de s'attarder. Par contre, l'idée de la conception qu'on peut se faire de l'informatique, à savoir l'idée qu'il suffit de mettre en place un beau système informatique pour régler tous les problèmes, mérite d'être analysée davantage.

Trop souvent, on tend à oublier qu'un système informatique n'est qu'un outil de travail. Un outil très puissant, mais dont la performance dépendra d'abord de ceux ou celles qui l'auront conçu, puis de ceux et celles qui l'implanteront et, finalement, des utilisateurs eux-mêmes. Avant même tout cela, la valeur d'un système dépendra de l'assurance que les données traitées par le système soient fiables, au départ. Souvent, on ne pense que trop tard aux problèmes de sécurité, de communication, de temps-réponse et de contrôle qui surgissent immanquablement, même dans les meilleures solutions.

---

[1] M. Parizeau est programmeur chez Mathema Inc., membre du groupe Sodarcan.

Autre point : supposons qu'une entreprise bien gérée ait déjà pris soin de toutes ces questions, qu'elle ait déjà dépensé une petite fortune en études de tous genres (ce qui n'est souvent pas le cas), que le système soit déjà installé et qu'il fonctionne. Mais fonctionne-t-il bien ? Répond-il aux besoins réels de l'entreprise, ainsi qu'aux nouveaux besoins qui ont pu surgir depuis les premières études de faisabilité, il y a deux ou trois ans ?

Dans cette perspective, l'article qui suit cherchera à tracer certains acquis, quant à ce que j'appellerai une bonne gestion de l'informatique.

La règle #1 : clarifier les mandats dès le départ

Un des principaux problèmes, en matière de gestion informatique, est le fait qu'au début d'un projet les mandats ne sont pas toujours clairement établis. Quels sont les objectifs ? Quelles seront les responsabilités de chacun ? Qui prendra les décisions ? Quelles étapes faudra-t-il suivre ? Quels impacts cela aura-t-il ? Quels seront les budgets ? Quand le projet devra-t-il être complété ? Répondre à ces questions devra toujours être le point de départ, car cela évitera bien des surprises, frustrations et malentendus. Cela assure en même temps, dès le départ, un appui ferme de tous les intervenants. Cela fait, ceux-ci devraient être conscients qu'il faudra nécessairement faire des compromis, donc des choix.

Comme informaticien, j'ai souvent rencontré des gestionnaires qui exigeaient que leur système informatique puisse tout à la fois répondre à tous leurs besoins, même les plus secondaires, être des plus faciles d'utilisation, réagir à toute demande rapidement et, bien sûr, sans que cela n'entraîne de délais de livraison supplémentaires. En même temps, il fallait réduire les coûts de développement ou d'opération jugés trop élevés.

Même si de tels objectifs, pris séparément, peuvent se justifier, pris ensemble, ces derniers entrent régulièrement en conflit. Pour solutionner les problèmes, il faut faire des choix. Malheureusement, ce n'est pas toujours facile.

La règle #2 : éviter les raccourcis

Lorsqu'on commence un nouveau projet, trop souvent on tend à privilégier les solutions de raccourci et celles qui, à court terme, pa-

raissent les plus économiques, sans se rendre vraiment compte qu'en informatique (comme dans la plupart des domaines, d'ailleurs), les raccourcis ne sont pas toujours les plus avantageux, à long terme.

Prenons l'exemple d'une compagnie, que nous appellerons ABC Inc., et qui veut informatiser son système de comptabilité. À cause de la complexité du système actuel, mais aussi à cause d'expériences passées négatives, ABC Inc. hésite à donner le contrat à son propre département de développement informatique. D'autant plus qu'il est déjà surchargé avec d'autres projets en cours. Peut-être
410 pourrait-on charger une firme extérieure spécialisée dans le domaine d'accomplir la tâche. Peut-être pourrait-on voir sur le marché si un logiciel existant pourrait satisfaire les besoins. Ce serait une autre solution. Ou peut-être encore pourrait-on donner le contrat, malgré tout, à son département de développement. Ou peut-être devrait-on tout simplement remettre à plus tard l'informatisation de sa comptabilité. Alors, que devrait-on faire ?

Si ABC Inc. est surtout concernée par l'élément temps et argent, l'option d'un logiciel déjà tout fait (*package*) pourrait être la solution. Par contre, il est presque assuré qu'il faudra changer certaines des procédures et des méthodes dans le département de comptabilité, car le logiciel ne répondra peut-être pas *complètement* à la situation. Dans un tel cas, l'entreprise devra être prête à s'ajuster. Et si ABC Inc. n'est pas prête à s'ajuster à certains standards, alors peut-être devrait-elle aller vers les autres options, même si elles peuvent s'avérer plus coûteuses.

L'option de sous-traitance permettra sans doute plus de flexibilité, tout en conservant un contrôle assez serré des budgets de développement (suivant les clauses de contrats). Par contre, si celui-ci inclut non seulement le développement, mais aussi l'entretien futur du système, l'option rendra nécessairement ABC Inc. dépendante de son fournisseur et pourrait coûter plus cher que l'option *développement-maison.*

L'option donnera la meilleure flexibilité et le meilleur contrôle, quant au contenu du logiciel, ainsi qu'à son évolution future. Par contre, une telle option exigera peut-être certaines priorités au sein du département informatique, afin de respecter les échéanciers. Dans un autre ordre d'idées, cette option pourrait favoriser le déve-

loppement, à l'interne, de nouvelles expertises qui resteront dans l'entreprise : un atout certain pour le futur.

Chose certaine, l'entreprise devrait éviter de prendre une décision à court terme, car la situation pourrait évoluer, ce qui rendrait cruciales certaines considérations auparavant secondaires.

En définitive, la meilleure décision pour ABC Inc. sera celle qui tiendra compte des priorités et des attentes non seulement actuelles, mais aussi futures de l'entreprise.

La règle #3 : essayer de prévoir l'avenir

À première vue, une telle règle peut paraître évidente, mais l'expérience prouve que les règles les plus évidentes sont souvent les plus difficiles à retenir.

À ce propos, je soulignerai un exemple qui m'est arrivé il y a quelques années, alors que je travaillais à un système de gestion. Préalablement au développement du projet, les gestionnaires pensaient avoir tout prévu ; mais ils avaient oublié de se pencher sur un aspect pourtant clé : la fiabilité des données devant être utilisées à l'entrée (*input*). Dans ce cas-ci, les données provenaient d'un autre système vieilli, géré par une autre compagnie et sur lequel on n'avait pas de contrôle. Le projet monopolisa deux ans de travail et fut finalement mis au rancart. Malgré une série d'ajustements en cours de route pour corriger les erreurs contenues dans les données devant servir d'entrée au système et qui avaient été détectées au fur et à mesure du développement, on n'arrivait pas à garantir un haut niveau de fiabilité des données à la sortie (*output*). La morale de l'histoire : plutôt que de vouloir à tout prix ce système en se disant qu'on corrigerait bien les problèmes à mesure qu'ils surgiraient, il aurait peut-être mieux valu passer plus de temps, dès le départ, pour bien identifier tous les problèmes actuels et potentiels. On aurait alors peut-être découvert à temps l'ampleur des problèmes de non-fiabilité des données de base. On aurait peut-être conclu que le projet était prématuré, dans le contexte où on aurait trouvé une solution. Chose certaine, on aurait épargné beaucoup de temps, d'argent et de frustrations pour tout le monde.

En informatique, il existe une règle de base à l'effet que si quelque chose peut aller mal, cela ira mal. Et, en corollaire, on pourrait

aussi dire que si quelque chose ne peut pas aller mal, cela ira mal quand même.

En conséquence, toute décision devrait toujours être longuement mûrie, surtout si elle implique d'importantes sommes.

Autre chose : on devrait toujours prendre garde aux solutions parfaites ou sans défaut. Comme je l'ai déjà dit, toute solution informatique exige des choix. Cela sous-entend qu'elle possède à la fois des avantages et des défauts. On devrait donc toujours être conscient non seulement des avantages de toute solution, mais aussi de ses dé-
412 fauts, surtout lorsqu'il s'agit d'une formule qui n'est pas développée à l'interne et sur laquelle l'entreprise aura généralement moins de contrôle.

À ce propos, une entreprise ne devrait jamais baser ses opérations informatiques sur des solutions qui ne sont pas au point ou dont l'espérance de vie n'est pas réellement assurée. À moins, bien sûr, qu'on soit d'accord pour jouer le rôle de cobaye. Là encore, cela peut avoir certains avantages à court terme, mais aussi de forts désavantages, à long terme.

La règle #4 : maintenir un contrôle serré sur toutes les étapes d'un projet

En 1983, dans l'État du New-Jersey, commença un projet pour refaire complètement le vieux système de gestion du Département des Véhicules Motorisés (DMV), qui se termina en 1985. Il coûta au gouvernement de l'État $15 millions U.S. Il représente un cas classique de mauvaise gestion informatique. Cité en exemple dans le numéro de février 1986 de la revue *Computer Decisions*, comme ce qu'il ne faut pas faire en matière de contrôle de projet, je pense qu'il serait intéressant de s'y arrêter un peu. Personnellement, j'en tire au moins deux leçons : d'abord, ne jamais mettre tous ses oeufs dans le même panier. Plus on répartit les responsabilités, plus il devient facile de contrôler chaque étape d'un projet. Ensuite, ne jamais remettre à plus tard une décision qui pourrait être prise maintenant, même si cela peut faire mal car, en la remettant, cela finit généralement par faire encore plus mal. Il n'y a rien de pire, en effet, que de laisser pourrir une situation.

Cela dit, revenons au cas de DMV.

Au départ, pour diverses raisons, l'administration du DMV ne croyait pas possible que le projet puisse être mené à terme à l'interne. À la place, elle privilégiait une firme extérieure, laquelle travaillait déjà pour DMV sur un projet séparé consistant à développer les précisions du futur système.

La maison obtint le contrat. Ce fut la première erreur, de l'avis de plusieurs des experts qui ont depuis analysé ce dossier. En effet, lorsqu'on donne un projet en sous-traitance, le groupe qui écrit les spécifications ne devrait jamais être celui qui développe en même temps le système.

DMV justifia, à l'époque, son choix par le manque de temps et les échéanciers trop justes. Ce qui devait arriver arriva. Les données de départ, incluant l'utilisation d'un langage de programmation dit de quatrième génération complètement inadéquat pour ce genre de projet, créèrent d'énormes problèmes de développement. De son côté, la maison qui avait un contrat à coût fixe, incluant de lourdes pénalités pour chaque jour de retard par rapport aux échéanciers, refusa de changer de cap jusqu'à la fin. De plus, ayant elle-même conçu les spécifications de départ, *** ne voulait probablement pas perdre la face.

DMV ne faisait pas confiance à son propre personnel et avait donné carte blanche à ***. Ce faisant, DMV s'était en même temps mis une corde autour du cou, qui ne devait pas tarder à se resserrer. Et ce fut la deuxième erreur.

Le résultat fut catastrophique. Le traitement de nuit prenait des jours. Un usager pouvait attendre une heure avant d'entrer dans le système. Six mois après la date prévue de livraison, celui-ci ne fonctionnait pas encore adéquatement. Entre-temps, l'ancien, qui aurait pu rester en fonction pour pallier à la situation, avait préalablement été supprimé (par excès de confiance ?). Toute l'administration du DMV fut chambardée pendant des mois. Des milliers d'automobilistes se promenaient avec des permis expirés qui ne pouvaient pas être renouvelés. Des centaines de milliers d'erreurs furent ainsi constatées.

Pendant des mois, DMV dut travailler pour essayer de réparer les pots cassés. Le personnel informatique de DMV, servant de « porte de sortie », fut mis à contribution. Une cinquantaine de pro-

grammes interactifs (sur un total de 800) furent refaits au complet, de même que tous les programmes pour le traitement de nuit. Le système fonctionne aujourd'hui. Cela comporte au moins une consolation, en même temps qu'un autre principe de base à ne pas oublier : comme quoi même la situation la plus désespérée peut parfois être corrigée, si l'on prend les moyens pour reprendre le contrôle d'une situation devenue incontrôlable.

La règle #5 : bien planifier l'introduction

Cela dit, même le meilleur système au monde ne fonctionnera pas nécessairement, si son introduction dans la vie de l'entreprise n'est pas bien planifiée. Au contraire, cela pourrait même être un échec cuisant.

À ce propos, je lisais récemment, dans un article du *Financial Post* du 18 mai 1987, intitulé "*Right answer goes wrong*", que jusqu'à 65% de toutes les tentatives d'implanter le système MRPII (pour *Manufacturing Resource Planning II*) dans les entreprises nord-américaines étaient un échec. Ce système, qui coûte $250,000, est considéré comme la fine pointe de l'informatique, en matière de contrôle et de planification, dans une entreprise manufacturière. Selon cet article, le problème central serait une mauvaise compréhension des changements de mentalité et de fonctionnement, qui doivent absolument être planifiés et introduits avant et pendant la mise en place du système.

Il est assez courant de négliger, lors de l'introduction d'un nouveau système, la planification des ressources humaines additionnelles, qui seront requises pour l'entrée des données de base dans le système et pour son démarrage. Souvent, on ne porte pas suffisamment attention aux réorganisations administratives qui devront peut-être avoir lieu. Même chose pour l'aménagement des lieux de travail, s'il s'agit de la première fois qu'on implante un système informatique dans un service.

Quand cela arrive, les utilisateurs ne sont alors pas préparés pour la venue du système. Parfois, celui-ci peut être mal vu par les utilisateurs, ce qui entraîne un blocage de leur part.

La planification de l'entrée d'un nouveau système devrait toujours tenir compte de l'aspect humain. On serait surpris de voir l'impact d'un mauvais éclairage ambiant ou d'un poste de travail mal

adapté à l'entrée de données ou d'une charge de travail trop lourde sur la perception qu'un utilisateur peut avoir d'un système très bon, au départ.

La règle #6 : maintenir une bonne communication avec les intervenants

Tout cela nous amène finalement à un dernier point : l'importance de la communication avec le personnel d'un projet. Dans n'importe lequel, il y aura toujours un certain nombre de frictions entre les intervenants. Frictions entre les gestionnaires du côté usager et les informaticiens ; frictions entre les utilisateurs et les gestionnaires ; ou même frictions entre les informaticiens eux-mêmes ou entre les gestionnaires.

Généralement, ces frictions proviennent du fait que chaque groupe considère que l'autre ne comprend pas ou ne veut pas contribuer à la solution de problèmes qui le touchent particulièrement.

Personnellement, j'aimerais suggérer le plus d'interaction possible entre tous ces divers groupes, ce qui favorise le travail d'équipe. Chaque fois que cela se fait, cela contribue généralement à instituer un meilleur respect et une meilleure confiance mutuelle : deux atouts majeurs pour la réussite de n'importe quel projet.

Conclusion

Pour conclure, je dirai qu'un projet bien géré devrait viser à inclure un engagement clair et précis de tous les intervenants, dès le départ, une bonne planification de toutes les phases d'implantation du projet, une analyse judicieuse des choix qui devront nécessairement être faits tout au long du travail et, enfin, de bonnes relations entre les groupes impliqués.