Chapitre 17 — Comment en est-on arrivé là ?

On l'a vu dans le chapitre précédent, le bilan du système d'information n'est pas des plus brillant !
Bien sûr, il serait absurde de noircir le tableau à outrance, on trouve toujours de brillantes exceptions. Cependant, dans l'ensemble, il vaut mieux reconnaître que le résultat n'était vraiment pas à la hauteur de tant d'années de développements et des moyens (financiers, techniques, humains) déployés. Alors, pourquoi ce bilan décevant ?
C'est que le système d'informations des années 1980 et 1990 était dépendant des techniques et pratiques en vigueur à ces époques, il a donc souffert des tares et des vices en vogue pendant toutes ces vingt dernières années d'informatiques…

Il faut revenir sur les bases conceptuelles qui sont à l'origine de bien des problèmes. La plupart du temps, les besoins des utilisateurs étaient mal compris, les applications nécessaires aux organisations étaient mal analysées et les projets pour les développer étaient mal gérés. Une grosse partie de ces mauvaises bases est imputable aux méthodes de conception, développement et gestion de projets qui étaient très populaires dans les entreprises pendant les années 1970 et 1980.

La démarche de développement "chaotique"

L'apparition des méthodologies de développement et de suivi de projet peut être vue comme une réaction à la démarche "chaotique" qui régnait dans les premiers temps de l'informatique moderne (la décennie 60). La plupart des activités de développement logiciel suivaient un cours chaotique souvent caractérisé par la phrase "coder et corriger". Le logiciel était écrit sans plan d'ensemble et la conception des systèmes était le fruit de nombreuses décisions basées sur le court terme (tout cela est encore vrai dans bien des cas…). Ce type de pratique peut convenir pour des petits programmes mais si on veut mettre en place des applications plus importantes, il devient de plus en plus difficile d'ajouter des fonctions au système au fur et à mesure de sa "construction".
De plus en plus de défauts deviennent proéminents et il devient de plus en plus difficile de les corriger.Une longue période de tests après que le logiciel soit considéré comme terminé sur le plan fonctionnel est un signe typique de ce type de démarche. Elle vient remettre en cause le calendrier du projet puisque la durée des tests et du debug est presque toujours impossible à évaluer (certains s'y essaient mais le taux d'erreur est alors proche de 100 % !).

Une alternative au chaos : les méthodes

Les organisations utilisatrices d'informatique ont dû faire avec ce style de développement pendant les années soixante mais une alternative est apparue au début des années soixante-dix : les méthodologies de développement et de suivi de projet. Les méthodologies imposent une discipline sur les processus de développement logiciel avec pour but de rendre le développement plus prévisible et plus efficace. Ce but est théoriquement atteint en suivant un enchaînement rigoureux de tâches et en respectant à la lettre un plan-

ning détaillé. Ces méthodologies informatiques sont inspirées ou même dérivées des pratiques connues dans d'autres disciplines faisant appel à l'ingénierie. En France, la méthode Merise était le symbole principal de cette démarche.

> === À propos de Merise
> Extraits modifiés à partir de http://merise.developpez.com/faq/?page=GENE#INTRO_Historique
> Issue de l'analyse systémique, la méthode Merise est issue des travaux menés par Hubert Tardieu dans les années 1970 et qui s'inséraient dans le cadre d'une réflexion internationale, autour notamment du modèle relationnel d'Edgar Frank Codd. Elle est devenue un projet opérationnel au début des années 1980 à la demande du ministère de l'industrie, et a surtout été utilisée en France, par les SSII de ses membres fondateurs (Sema-Metra, ainsi que par la CGI Informatique) et principalement pour les projets d'envergure, notamment des grandes administrations publiques ou privées.
> Merise voit officiellement le jour en 1979, sous la forme d'un premier fascicule publié par Ministère de l'Industrie : "Méthode de définition d'un système d'information". Le nom de Merise a été trouvé comme la métaphore du merisier qui doit être greffé pour porter des fruits. En effet, dans l'introduction du premier fascicule, il est écrit : « Merise n'est pas une méthode, mais un tronc commun méthodologique…. ». En effet, les différentes SSII veulent pouvoir ultérieurement apporter leur valeur ajoutée en personnalisant cette méthode. Le vers est dans le fruit. Quinze ans après, on observera des méthodes Merise, parfois avec des écarts notables. Par ailleurs, quasiment aucune structure (tel l'OMG avec UML) ne veillera au respect d'une certaine "normalisation" de Merise.
>
> Le projet Merise se poursuit donc jusqu'en début 1981 avec la publication de plusieurs documents de référence sur la méthode Merise. À partir de 1981, certaines grandes SSII qui avaient accompagné Merise, dont SEMA, CGI, GAMMA [devenu depuis MEGA = MEriseGAmma], entament la diffusion de la méthode auprès des grandes entreprises et de l'Administration. En 1983, est publié le premier ouvrage sur Merise, ouvrage qui restera la référence : "La méthode Merise – Tome I : Principes et outils" de H. Tardieu, A. Rochfeld, R. Colletti. Suivi en 1985 par : "La méthode Merise – Tome II : Démarches et pratiques" de H. Tardieu, A. Rochfeld, R. Colletti, G. Panet, G. Vahee.
>
> Entre-temps, l'Administration consacre Merise comme la méthode de référence pour tous ses projets de conception de SI, assurant ainsi sa pérennité et son enracinement. Dès lors, Merise connaît un engouement. Quasiment toutes les SSII font désormais du Merise (avec des résultats plus ou moins probants…). De nombreux ouvrages paraissent. Merise est désormais enseigné dans les formations universitaires. Des outils apparaissent, certains issus des prototypes conçus lors de la recherche. La fin des années quatre-vingt dénombrera plus de 15 outils français sur Merise. Quasiment toute grande SSII propose le sien. Certains sont encore sur le marché. En 1990 Merise est devenu une figure imposée dans le cursus de formation de tout informaticien, du moins sur la partie de modélisation, plus particulièrement des données.
> ===

Trop lourd pour être efficace

La critique la plus fréquente qui est faite à l'encontre de ces démarches c'est qu'elles sont bureaucratiques. Il y a tant à faire pour simplement suivre la démarche que le rythme des projets s'en trouve ralenti. Du coup, on les appelle souvent les méthodologies lourdes, voire "monumentales"… Les méthodes monumentales sont, le plus souvent, basées sur un gros classeur où tout est décrit et où tout doit être consigné. Ces méthodologies sont connues et utilisées depuis le tout début des années quatre-vingt. Le moins que l'on puisse dire est qu'elles n'ont pas donné de résultats spectaculaires et, conséquence logique, elles sont devenues de moins en moins populaires au fil des années (mais elles ont tout de même eu le temps de produire des dégâts…). Ces approches étaient surtout conçues pour que les intervenants de base sachent au moins où il fallait poser les pieds pour avancer et ainsi donnent l'illusion de savoir ce qu'ils faisaient…

Le désaveu qui a finalement frappé les méthodes traditionnelles s'explique facilement : ces démarches strictes sont sans doute adaptées aux domaines de la fabrication en série dans un contexte industriel mais pas pour le développement logiciel qui est très éloigné du systématisme qui a cours dans l'industrie. L'ère industrielle a été l'avènement de la prédictibilité en mesurant le plus que l'on pouvait : "comment produire tant de pièces avec tant de ressources, tant de personnel, et avec un taux de rejet de moins de tant de %". Or ce modèle ne fonctionne pas en informatique qui reste encore sujette à trop de fluctuations. En particulier, l'ère industrielle a voulu que tous les employés aient le même rendement — et soient donc facilement interchangeables voire même remplaçables. En informatique par

contre cela veut souvent dire un nivellement par le bas, surtout que c'est un milieu où, contrairement à une chaîne de montage, un seul employé vedette peut être plus productif (et ce de plusieurs ordres de magnitude !) qu'une armée d'employés médiocres.

L'absence de progrès palpables…

Pendant des années, le domaine des méthodes informatiques a été agité par une activité soutenue : les modes s'y succédaient, les gourous pullulaient et les débats faisaient rage. Les clients y croyaient et n'hésitaient pas à dépenser des sommes importantes en outils et en séminaires. Ce domaine s'est même donné un "nom de scène" qui sonnait bien : le génie logiciel.
Au temps des premiers pas du génie logiciel, l'idée était simple et élégante : créons des logiciels qui nous aident à écrire des logiciels. Mais, rapidement, les choses sont devenues incontrôlables : les outils sont devenus trop lourds et trop contraignants pour apporter une aide réelle dans les projets de développements. Ces outils qui avaient été créés à l'origine pour aider les informaticiens à suivre les règles sont devenus trop difficiles à utiliser par eux-mêmes. Les développeurs d'applications trouvèrent nécessaires de pratiquer des raccourcis et même "d'oublier" des pratiques importantes, simplement pour respecter les délais des projets dans lesquels ils étaient engagés.

=== À propos de la "balle d'argent"
"Pas de balle en argent" (traduction littérale de "No Silver Bullet") est une expression introduite en génie logiciel dans les années 1960 par Frederick Brooks lorsqu'il a publié No Silver Bullet — Essence and Accidents of Software Engineering. Brooks désigne ainsi l'ensemble des "techniques miracles" censées permettre magiquement d'augmenter la productivité des programmeurs et de diminuer la quantité de bugs dans les programmes produits, et ainsi de tuer le monstre redouté, le dépassement des délais, lors de la réalisation des projets informatiques. L'expression est un jeu de mots entre d'une part le fait que dans une présentation, les puces au début de chacune des phrases s'appellent en anglais bullet (s), d'autre part le fait qu'une balle d'argent est, dans les légendes, le seul projectile capable d'abattre un lycanthrope, et a donc le statut d'arme miraculeuse.
Brooks, qui a relaté son expérience dans Le Mythe du mois-homme, a par la suite écrit un article marquant, No Silver Bullet, où il met en doute les "technologies miracles" de son temps. L'expression Silver Bullet est depuis entrée dans le langage du génie logiciel. L'opinion de Brooks est que les difficultés de réalisation des logiciels se divisent en difficultés accidentelles (langages de programmation et systèmes laborieux et malaisés à utiliser) et en difficultés essentielles (inhérentes à la production de logiciels). Or, selon lui, les difficultés accidentelles ont déjà été en grande partie éliminées, par exemple par l'adoption de langages de haut niveau ; il n'y aura donc pas dans le futur de nouveaux progrès permettant de gains importants de productivité. Il cite ensuite un certain nombre de technologies présentées comme devant révolutionner l'industrie logicielle (le langage Ada et la programmation orientée objet) et explique que si ces technologies permettent d'encore diminuer les difficultés accidentelles de la programmation, elles ne peuvent en supprimer les difficultés essentielles.
===

La nostalgie de l'âge d'Or des méthodes

On ne peut pas se souvenir sans une certaine nostalgie de ces vagues successives : d'abord l'approche conceptuelle (avec Merise côté français mais les autres pays ont aussi eu droit à la dictature des méthodes basées sur le couple "entités-relations"), la mode du CASE, l'espoir mis dans le développement articulé autour d'un "référentiel" (ah, AD Cycle et le Repository, qu'êtes-vous donc devenus ?), ou dans les diverses approches liées à l'objet (où l'on avait promis des gains formidables en productivité… Que l'on attend toujours !), puis le RAD. Avec le RAD (pour Rapid Application Development, approche de développement rapide à base de prototypes itératifs, le terme RAD a été inventé par James Martin en 1991), on a bien senti que c'était le "commencement de la fin" pour les méthodes car il s'agissait enfin d'une démarche orientée résultats (quelle horreur !).
Aujourd'hui, toute cette effervescence est bien retombée, faute de bénéfice, on a enfin fini par comprendre que finalement, non, il n'y avait pas de "balle d'argent" !

=== À propos du prototypage
De même que les industriels commencent toujours par procéder à un prototypage avant de construire de coûteuses usines afin de mettre en évidence les défauts qui n'avaient pas été imaginés, il est conseillé au concepteur de logiciels de réaliser une maquette, ou un prototype (ou les deux)