



**REENGINEERING**  
des développements informatiques

**Vite fait,  
bien fait !**

**Le paradigme  
du futur immédiat**

*Jean-Pierre Vickoff*

Dépôt Février 98

# REENGINEERING

## Une opération vitale

L'informatique évolue et les applications se sophistiquent autant que leur typologie se diversifie. L'échec de projet devient une hantise justifiée par les statistiques autant que par les expériences fâcheuses.

La première mutation fut celle de Windows et du Client-Serveur classique. L'émergence du NET (Internet / Extranet / Intranet) et des architectures multicouches consomment la rupture avec les anciens modèles et outils de développement.

L'informaticien réalise alors que les techniques de conception, de planification et de réalisation d'un système opérationnel lourd ne peuvent s'appliquer sans danger et sans déperdition de ressources à des applications décisionnelles, stratégiques ou " Kleenex ", car en effet, il s'avère catastrophique de :

- modéliser
- standardiser
- développer
- sécuriser
- documenter

à l'identique, ces diverses catégories d'applications.

Dans un registre plus fonctionnel et économique, la recherche permanente de qualité que nécessite la différenciation concurrentielle est au cœur des préoccupations de l'organisation. La finesse et la complexité des nouveaux besoins implique alors l'engagement d'un utilisateur averti et motivé autant que responsable et demandeur.

Pour répondre à ces exigences de services, l'informaticien comprend qu'il doit s'impliquer dans une ré-ingénierie de la fonction développement d'application et qu'il lui faut disposer d'un nouveau cadre méthodologique :

- puissant mais ouvert
- offrant la capacité de :
  - s'adapter à la typologie des applications
  - de se dimensionner en fonction des contraintes (budget, délai, qualité)
- prenant en compte la dimension « communication interpersonnelle »

Pour résumer, il faut adapter une méthode " à la carte " tout en maîtrisant les conditions d'un accompagnement fiable et déterministe. Ainsi se décline les conditions actuelles de la réussite du développement d'applications. Une des solutions : le RAD.

## **Le RAD <sup>®</sup> ne préfigure aucun miracle**

Le RAD ne permet pas miraculeusement de développer plus vite. Pas plus qu'un Grand Prix ne se gagne par hasard.

Le RAD est une compétition de hautes technologies. Il demande une volonté d'investir, de l'organisation, des moyens techniques optimal, le sens du risque calculé, un personnel spécialisé, formé, entraîné et motivé.

Alors ensuite, et seulement ensuite, vous obtenez la performance, la réussite et un retour sur investissement exceptionnel :

- une application totalement approuvée répondant aux besoins des utilisateurs ;
- la sécurité d'une visibilité qualitative et quantitative de l'avancement des travaux.
- la livraison rapide des fonctionnalités stratégiques ;
- des coûts de développement permettant un retour sur investissement concret ;

Le RAD, (comme Réelle Approbation des Développements par les utilisateurs ?) représente en fait l'état de l'art en matière de qualité des procédés de développement d'applications informatiques. RAD2 intègre votre savoir faire actuel et reste ouvert sur les évolutions futures (CMM, UML, ... ).

*Cette publication est une introduction à :*

**REENGINEERING RAD, CMM, UML  
« Conduite de projet, La Pratique »**

**REENGINEERING, RAD2  
« Le Processus qualité, La Mise en œuvre »**

*et un complément de :*

**RAD « Développement rapide d'applications »  
aux éditions Macmillan (en librairies)**

*Ces ouvrages sont détaillés dans les dernières pages, ainsi que divers outils. Ils peuvent être commandés économiquement sur un CD à partir du site WEB*

**HTTP ://WWW.RAD.FR/BONCDE.HTM**

# Table des matières

<b>1. Projets informatiques, échecs: 84%</b>	<b>8</b>
1.1 Une vision du stratégique en péril	8
1.2 Les enjeux du reengineering	11
1.3 Technologies, relations humaines et méthodes	12
1.4 RAD un des piliers du reengineering	16
<b>2. Le reengineering s'impose ?</b>	<b>18</b>
2.1 Un phénomène violent	18
2.2 Principes fondamentaux	18
2.3 Synthèse / phasage d'un projet moderne	20
2.4 Votre projet est-il « RAD » ?	24
2.5 Direction de projets stratégiques	26
2.6 Qualité et motivation des hommes	27
2.7 Evolution du RAD et du reengineering	34
2.8 Causes de rejets	34
2.9 Histoires d'aberrations	35
2.10 Conditions d'échec	36
2.11 Quels projets pour le RAD ?	38
2.12 Les nouvelles applications	38
2.13 Organisation existante et reengineering	40
2.14 Puissance et simplicité du RAD	41
2.15 La méthode Merise et le RAD	42
2.16 Difficulté de l'animation de projet	43
2.17 Culture française et reengineering	44

2.18 Les pompiers du reengineering	45
2.19 Evolution et causes de rejet	46
2.20 Le besoin du client (interne ou externe)	47
2.21 Une vision globale du développement	48
2.22 Histoires d'architecture technique	49
2.23 Histoires d'outils non appropriés	50
2.24 Méthode, bon sens et reengineering	51
2.25 Les raccourcis méthodologiques	52
2.26 Histoire de conception inadaptée	53
<b>3. Pour une performance indispensable</b>	<b>58</b>
<b>4. Annexes : CMM &amp; UML</b>	<b>62</b>
4.1 Le CMM en résumé	62
4.2 UML : utilité pour un gestionnaire	64
<b>5. Bibliographie des ouvrages</b>	<b>67</b>

# Echecs et méthodes

*« Ce n'est pas la technique qui représente le vrai danger pour la civilisation, c'est l'inertie des structures. »*

*Louis Armand (1905-1971), Plaidoyer pour l'avenir*

## 1. Projets informatiques, échecs: 84%

### 1.1 Une vision du stratégique en péril

Les développements informatiques constituent désormais l'activité humaine la plus récente, la plus complexe et le facteur de progrès le plus important qui conditionne la croissance, la mutation et la survie de la plupart des secteurs de l'économie moderne.

Insatiable, incontournable, l'informatique s'est donc imposée comme outil stratégique aux chefs d'entreprise, mais la plupart ignorent que 53% des projets dérivent ou ne satisfont pas les utilisateurs et que 31% sont abandonnés avant leur mise en exploitation (sources 1996 : Standish Group et Sema Group).

**Ce gouffre économique est la conséquence directe du manque de concertation entre informaticiens et utilisateurs, de l'inadaptation des méthodes à l'évolution technologique et de la durée excessive des projets.**

Ces chiffres sont édifiants mais non surprenants. Une pratique de trois décennies de conception et de réalisation à titre de consultant permet d'acquérir une large vision du sujet. Durant toutes ces années, aucune des organisations visitées ne disposait d'un processus qualité formalisé, détaillé, complet et spécialisé pour guider ses développements. Au mieux, une approche de modélisation comme Merise ou une méthodologie de conduite de projet proposaient quelques grandes lignes directrices. Les statistiques résultantes de ces pratiques sont éloquentes quant à leur efficacité et à leur effectivité, ce qui n'est pas la même chose.

**Lorsque vous achevez une application dans le temps planifié, c'est peut-être de l'efficacité, mais pas obligatoirement de l'efficience. Si les attentes de l'utilisateur ne sont pas « rencontrées », ce n'est certainement pas de l'effectivité.**

L'efficacité caractérise le degré de réalisation des objectifs. L'efficience introduit un critère d'optimisation des ressources employées. L'effectivité fait entrer en ligne de compte la notion de pertinence.

L'histoire des développements informatiques est celle d'une série de catastrophes le plus souvent bien dissimulées. Peut-on imaginer construire une voiture, un avion ou un immeuble, sans plan, sans prototype et sans chaîne de montage ? En développement informatique, la chose est tentée chaque matin. Le constat statistique de 84% d'échecs relatifs ou totaux met en évidence la réalité d'un problème que la

complexité des applications et des technologies amplifie dramatiquement. Des centaines et des milliers d'années d'ingénieurs ont été ainsi englouties en pure perte dans des projets pharaoniques aux frontières du réel. Une réingénierie des procédés de développement s'impose.

Le problème est lié à la fois à de nouveaux types d'applications (décisionnelles, stratégiques), à un accroissement de la complexité de l'environnement de réalisation (relationnel, client-serveur, Windows, Workflow, Groupware, Internet, etc.) et surtout à l'absence d'une méthode simple et adaptée comme le RAD (*développement rapide d'applications réellement approuvées par les utilisateurs*).

**Une approche simple et efficace donne à l'organisation les moyens d'obtenir une réactivité dont le profit stratégique permet ensuite d'accroître les bénéfices parallèlement à la satisfaction du client.**

La plupart des informaticiens classiques et des gestionnaires de l'informatique sont totalement déconnectés des contraintes réelles impliquées par ces évolutions. C'est pourtant la survie même de beaucoup d'organisations, et donc notre niveau de vie, qui dépend de la maîtrise concrète de ces techniques.

Peut-on en déduire que la conception est en crise ? Entendez par « crise » : incapacité d'introduire un processus d'adaptation permanente aux évolutions rapides et globales des besoins et des technologies. Mais de quelle conception est-il question ? L'informatique ne s'apparente pas à de la poterie ou à de l'artisanat. Une application actuelle est devenue un assemblage de milliers de fonctions, d'objets, de composants. Les champs d'application se sont élargis et supportent des concepts complexes allant jusqu'à couvrir la totalité des activités d'organisations sophistiquées. Ces deux dernières décennies, la bonne santé financière des organisations et l'incapacité pour des dirigeants de chiffrer réellement les coûts (directs et avantages concurrentiels non obtenus) ont évité de trop pénibles remises en question.

Les chiffres et les faits parlent d'eux-mêmes, il y a effectivement crise, mais elle ne se limite pas à la conception. C'est toute l'organisation des développements qui est concernée et en premier lieu les méthodologies de conduite de projet qui n'ont pas su s'adapter. Il faut maintenant prendre en compte simultanément la réduction des cycles de vie, l'accroissement de la complexité applicative, l'impact du tout « Windows », des outils « Visual », la stratification des architectures client-serveur, les bases d'informations distribuées, l'émergence d'une informatique « nomade » et la disparition de l'approche systémique au

profit d'une vision axée sur le flux qualité. La fin des dinosaures est proche ; le progrès et l'efficacité ne connaissent ni limite ni frontière : mon stylo est *made in Taiwan*, et mon ordinateur *made in USA*.

La crise du développement résulte donc des nombreuses évolutions dont les impacts n'ont pas été intégrés progressivement aux méthodes, aux techniques et aux outils. La notion globale de conception repose pourtant sur la maîtrise simultanée de l'ensemble de ces éléments. Parallèlement à ces aspects techniques, une autre « crise » réduit de manière drastique les budgets consacrés à cette activité. Oui, la conception est confrontée à une crise, et il était temps qu'elle le soit.

Heureusement, en réponse, émerge une approche complémentaire axée sur la qualité et la communication. Elle s'appelle RAD, RAD comme radicale ou RAD comme Réelle Approbation des Développements. Le RAD représente l'état de l'art en matière de qualité des procédés de développement d'applications informatiques. Le RAD est très apprécié du monde anglo-saxon, où son efficacité a fait ses preuves. L'objectif principal de cette méthode est la maîtrise de la qualité. Accessoirement, elle réduit les délais de production et facilite la maîtrise des coûts. Malheureusement, la France intègre mal cette culture et oppose même une vive résistance.

Ainsi, certains se contentent de coller l'étiquette RAD sur leurs habitudes et feignent d'être surpris de ne pas obtenir ensuite les résultats escomptés.

Beaucoup d'autres refusent cette évolution malgré les échecs répétitifs de leurs certitudes. Mettre en œuvre le RAD requiert donc courage et motivation ou bien restriction budgétaire et obligation de résultat. Idéalement, le processus devrait être initié conjointement par les directions opérationnelles et la direction informatique sous l'égide d'un « facilitateur » neutre répondant à la direction générale.

Dans un contexte économique qui se radicalise, l'informatique est devenue l'outil de la compétitivité. La réussite des projets est désormais vitale. Les principes du RAD seront donc acceptés. Espérons que le réveil imposé par cette réingénierie des procédés de développement ne sera pas trop difficile pour certains et trop tardif pour les autres.

## 1.2 Les enjeux du reengineering

Parfois, la réussite d'un projet informatique se matérialise par :

- une application totalement approuvée répondant aux besoins des utilisateurs ;
- la livraison rapide des fonctionnalités stratégiques ;
- des coûts de développement permettant un retour sur investissement concret ;
- la sécurité d'une visibilité qualitative et quantitative de l'avancement des travaux.

**Pour un professionnel de l'informatique, il n'y a pas de miracle ni de circonstance, seulement des techniques de réduction du risque : phasage, dimension temporelle, validation permanente, jalons zéro défaut, focus de visibilité.**

Il a été constaté que les méthodes de conduite de projet dites classiques, c'est-à-dire basées sur la validation de documents rédigés par des analystes, puis confiés à des programmeurs, mènent désormais le plus souvent à un échec dans les projets modernes et complexes.

Les différences fondamentales entre une conduite de projet classique et une conduite de projet moderne sont les suivantes :

- La **structuration des phases** est différente. C'est particulièrement évident pour la réalisation qui applique les techniques du prototypage et impose une spécification détaillée directement exprimée en termes de code.
- La **dimension temporelle** est différente, les projets se planifient en lots de 120 jours maximum. D'autre part, la réalisation représente désormais 50% de la charge.
- Les **intervenants** sont différents. Les fonctions d'analystes « pondant du dossier » pour des programmeurs « pondant du code » s'estompent. Un profil unique de concepteur-développeur hautement compétent et motivé s'avère indispensable. Les équipes rétrécissent, et le statut de « chef de projet » laisse place à une fonction coordination assumée généralement par un des membres du SWAT (équipe de projet spécialisée).

- L'**engagement des utilisateurs** est stable et permanent. Il est à la base du principe de validation permanente et de FOCUS qui garantissent visibilité et qualité.

Il découle de ces différences que les formes et les techniques de modélisation et de documentation doivent s'adapter tant à la méthode en elle-même qu'aux nouveaux besoins propres aux développements modernes.

Face à des développements de plus en plus complexes, la modélisation doit être approfondie, structurée, puissante et simplificatrice.

## 1.3 Technologies, relations humaines et méthodes

Pour l'organisation moderne, l'architecture client-serveur s'avère le principal facteur structurant de la décennie. Parallèlement, l'intégration technologique caractéristique de ce modèle ainsi que les contraintes de temps et de coûts bouleversent la conduite des projets informatiques. **Il faut livrer, même en fonctionnalités réduites, pour assurer un avantage immédiat ou un retour sur investissement accéléré.** Confrontées à ces défis, les méthodes traditionnelles s'enlisent et laissent le champ libre au *Développement Rapide d'Applications (RAD)* qui consacre la nécessité d'une totale synergie entre méthodes, techniques de communication, ressources humaines et outils.

### 1.3.1 Sur le plan des communications

A toutes les phases du développement, le RAD introduit la spécialisation et la gestion des relations interpersonnelles. Le classique chef de projet fait place à un binôme composé d'un coordonnateur fonctionnel issu de la maîtrise d'ouvrage qui impose la dynamique applicative et d'un coordonnateur technique représentant la maîtrise d'œuvre qui intègre la dimension technologique. **Les rapports entre les partenaires du développement se complexifient et se contractualisent.**

Le RAD implique alors un troisième groupe d'acteurs neutres organisé autour d'un animateur dont la mission est d'obtenir un consensus et une expression formelle des besoins par la maîtrise des techniques d'entretien, de prévision et de gestion des conflits. Des auxiliaires réalisent en direct la modélisation de l'application et sa validation lors de travaux de groupe organisés dans un espace de communication dédié (atelier de génie logiciel sur micros, moyens électroniques de rétroprojection, etc.).

### 1.3.2 Dans le domaine des ressources humaines

**Le prototypage déporte les spécifications détaillées dans la phase de réalisation. Les compétences de l'analyste et du programmeur fusionnent et imposent une mutation des métiers.** En parallèle, la pluralité technologique requiert une spécialisation accrue de l'équipe de projet. C'est le principe du *SWAT (Skilled With Advanced Tools)*. Un SWAT est composé de personnels de type spécialiste-généraliste, experts sur des techniques ou outils précis et nécessairement généralistes sur les autres. La coordination harmonieuse de ces profils à travers un contrat de projet initie une dynamique de groupe basée sur la gestion de la complémentarité. Cette synergie favorise le sens de l'identité de l'équipe, le partage de l'information et la notion d'entraide.

### 1.3.3 En termes de méthodologie

Jusqu'à maintenant, les méthodes se distinguaient par leur approche monolithique des problèmes. On décomposait la structure (*top-down*) ou l'on élaborait à partir des besoins (*bottom-up*). Une méthode de conduite de l'évolution et le RAD réconcilient les deux tendances. **La systémique s'applique à la conception haute et le prototypage à la spécification détaillée.** L'organisation est donc appréhendée par une approche qui garantit la cohérence systémique. C'est lors de la phase de réalisation que le prototypage s'impose pour valider l'adéquation du produit aux exigences des utilisateurs.

Concevoir nécessite de la méthode. Dans ce domaine, la décennie 90 a été marquée par quatre tendances parfois complémentaires et parfois structurellement incompatibles :

- 1) Empirique : Approche fonctionnelle (par les besoins) ;
- 2) MERISE : Approche systémique (par la structure) ;
- 3) OBJET : Approche mécanique (par les éléments de la structure) ;
- 4) RAD : Approche mixte et opportuniste (par le bon sens).

Actuellement, le discours de la méthode reste flou. Les adeptes de Merise voient dans l'objet ou dans le RAD un prolongement de leur savoir. L'objet ne voit que lui-même. Le RAD passe inaperçu, et les empiristes portent des lunettes noires. Pour ma part, je considère qu'il n'est pas plus aisé de voir de l'objet dans Merise, ou l'inverse, que de vouloir observer simultanément par les deux bouts de la lorgnette. De même, si le RAD

peut s'appuyer sur certains éléments de ces approches, l'inverse n'est pas inclus dans leur mode d'emploi. Analysons l'offre approche par approche.

Une approche exclusivement par les besoins (*bottom-up*) a l'avantage d'offrir une réponse immédiate à un utilisateur confronté à de multiples problèmes qui souvent le débordent. L'organisation travaille au coup par coup sans disposer de recul pour élaborer un projet guide. Il en résulte le plus souvent, outre le gaspillage de ressources, une multiplicité d'applications partiellement redondantes sans intégrité référentielle. L'aboutissement est en général la perte de contrôle du système d'information ou le déploiement d'efforts démesurés pour maintenir un semblant de cohérence.

La première réponse fut donnée dans la rigueur par les approches systémiques. Merise en France en est le meilleur exemple. La sécurité recherchée s'exprime par de multiples visions du système d'information à travers ses niveaux d'abstraction « naturels ». La contrepartie est l'accroissement du temps de conception. Dans les méthodes basées sur des cycles de validation en cascade, il faut en effet prendre le temps de formaliser chaque niveau et ensuite celui de les contrôler les uns par rapport aux autres. Dans le cas de modifications, d'omissions ou de compléments, naturellement induits par la durée de ces opérations, il est de plus nécessaire de procéder à de nombreux aller-retour dans le cycle de production.

En 1997, l'objet devient mature, se standardise et fédère ses prophètes. Cet aboutissement se concrétise par la fusion des approches de Grady Booch, d'Ivar Jacobson et de Jim Rumbaugh dans UML (Unified Modeling Language). **UML est une formulation unique de modélisation qui pourrait à l'avenir s'imposer comme standard de notation** (introduction à UML en annexe)..

L'objet en analyse et en conception représente pourtant un paradoxe. Indispensable pour la fabrication des éléments de base de nos applications, on peut douter de la capacité actuelle de ses outils (AGL) à appréhender de vastes systèmes de gestion. En première analyse, l'objet s'assimile à une approche hybride. Elle ne se base pas sur les besoins mais sur une décomposition, suivie d'un regroupement des éléments caractéristiques du système d'information. Pourtant, concevoir l'objet comme une approche méthodologique globale semble découler d'une simplification qui rapporterait l'architecture d'un immeuble à la somme des briques, des portes et des fenêtres, le composant. D'autre part, l'objet, comme aboutissement total, ne trouve parfaitement sa place que dans l'uniformité d'un « Meilleur des Mondes » dont personne ne souhaite

réellement la venue. Utilisons donc aujourd'hui l'objet pour ce qu'il est encore : un objet.

**Le RAD est un phénomène récent, à la fois cadre méthodologique, méthode de conduite de projet, outil de communication, technique de qualité et choix de modélisation.** Opportuniste et prudent, le RAD s'appuie d'abord sur une approche systémique (*top-down*) dans la première partie de sa mise en œuvre : le Cadrage. Cette phase correspond à une analyse, étendue aux aspects organisationnels, technologiques et financiers. En ce sens, certains aspects de Merise, une fois adaptés ou allégés, lui conviennent très bien .

Lors de la seconde partie d'un projet RAD : le Design, l'approche objet est mise à contribution avec pragmatisme afin de structurer et de pouvoir paralléliser la conception. L'apport de l'objet réside alors dans la parcellisation naturelle qu'il implique et dans l'isolation et l'homogénéité qu'il impose en termes de fonctionnalité. En ce sens, il ouvre la porte à la réutilisation et à l'industrialisation des composants « métier ».

La troisième partie : la Construction, s'appuie exclusivement sur le prototypage (*bottom-up*). A travers diverses techniques, le RAD plonge alors au cœur des besoins pour construire une application répondant à tous les critères de qualité et de conformité fonctionnelle. Et ce, jusque dans le moindre détail des attentes d'un utilisateur, participant actif depuis le début du projet. Dans cette phase, l'objet technique (OCX) trouve de nouveau sa place en fournissant les briques qui accéléreront la Construction.

#### 1.3.4 Vous avez dit « maturité » ?

Cet historique permet de comprendre le dilemme actuel du concepteur-développeur face à la disparition des assises sécuritaires méthodologiques qui jusqu'ici le soutenaient dans ses choix. **Que faire lorsque la méthode qui garantissait la réussite conduit au retard, à la dérive et à l'échec ?** La réponse n'est pas simple. S'adapter ne suffit pas et implique une forme de soumission, de réflexion de deuxième ordre. Par étapes, l'organisation doit s'élever vers la compétence maximum. Au préalable, il faut qu'elle soit en mesure de s'évaluer et qu'elle accepte de le faire. CMM (Capability Maturity Model) est le standard des outils répondant à ces besoins, il est de plus spécifiquement dédié au développement. C'est par la mise en œuvre de ce type d'outil que l'informatique sortira de la crise (introduction à CMM en annexe). La grande force de CMM est de permettre des paliers dans la rigueur et de ne pas reposer sur le principe du tout ou rien qui conduit trop souvent à rien.

Le pragmatisme du RAD consacre de fait la rupture entre la lenteur des cycles traditionnels et le dynamisme de la recherche d'une réponse optimale à des besoins en évolution. La précision de l'estimation et la possibilité de comparer instantanément des scénarios complexes acquièrent une importance vitale alors que le nombre de paramètres s'accroît. Il émerge donc de nouveaux types d'outils dédiés à l'évaluation (Évaluateur). Les phases de développement s'agrègent et deviennent plus denses. Elles sont instrumentées par des AGL puissants, légers, conviviaux, ouverts, adaptables au problème, modulables en fonction du contexte et de la taille du projet.

### 1.3.5 Vers le *high-tech* / *high-touch*

La direction du projet évolue vers l'adaptation permanente avec le mariage du *high-tech* et du *high-touch*. En Amérique du Nord (Bell, Abbot, Hydro-Québec) comme en France (Seita, Société Générale), les missions menées suivant les concepts du RAD (*time boxing*, *design to cost*) ont été conclues avec succès alors que la plupart des projets conventionnels dérivait hors des attentes des utilisateurs. Sans les avancées technologiques des cinq dernières années, l'émergence de cette méthode n'aurait pas été possible.

## 1.4 RAD un des piliers du reengineering

Nous devons les fondements du RAD à James Martin. Il publia le premier ouvrage traitant du RAD en 1991 aux Etats-Unis. Il ne fut malheureusement jamais traduit, ni diffusé à grande échelle en France, ce qui est certainement la cause de nombreuses incompréhensions et déviances dans les principes que l'on observe actuellement. Le premier ouvrage français traitant du RAD en France est *RAD, Le Développement Rapide d'Applications* de Jean-Pierre Vickoff déposé en 1995 sous le numéro Y-4542 à la Société des gens de lettre (SGDL). Il fit l'objet de publicités dans les principaux magazines et fut acheté, dès 1995, par les « grands comptes » (EDF, le CID, La Poste, GIAT, France Télécom, la Macif, etc.). Plus d'un an après, fin 1996, trois autres ouvrages traitant du RAD parurent certains dénaturant totalement la méthode. Certes, le RAD doit évoluer, mais pas sous la forme d'altérations opportunistes. Modifier la structuration des phases pour se rapprocher de Merise, ne pas comprendre la disparition des spécifications détaillées à travers le prototypage, et surtout altérer la dimension temporelle en rallongeant la durée des « time-box », sont des régressions et un reniement de l'héritage de James Martin.

# Le reengineering s'impose

*« Sachons que ce que nous venons de suggérer ...  
se fait déjà ailleurs ! »*

*Georges Archier/Hervé Sériex, Pilotes du 3<sup>ème</sup> type*

## 2. Le reengineering s'impose ?

### 2.1 Un phénomène violent

Nous vivons dans un monde violent. Aussi, le reengineering est-il un phénomène violent. Violent comme la guerre que mènent nos sociétés sur le plan commercial. Si vous en doutez, questionnez sur l'universalité des droits des exclus du travail et les exclus tout court. Cette violence, qui nous semble nouvelle, correspond à un rééquilibrage mondial entre les pays les plus pauvres et les plus riches. Dans ce grand mouvement de vases communicants un pays comme la France a beaucoup à perdre. La seule question est de savoir jusqu'à quel point, comment et en combien de temps. Le reengineering découle naturellement de ce monde conflictuel, le RAD est une de ses méthodes.

**Le RAD est une réponse performante mais contraignante à mettre en œuvre. C'est à la fois une culture et un investissement qui nécessite une évolution violente de l'organisation.**

Cet état de chose est à la fois la raison qui explique le rejet du RAD par les organisations qui ne sont pas encore acculées à la performance et son acceptation inconditionnelle par celles qui le sont. Le RAD s'affirme donc comme une vague de fond portée par la nécessité absolue de développer sous contraintes de temps, de budget, de qualité et de visibilité. Dans l'esprit de certains, le RAD se limite à du prototypage. Cette vision est dangereuse car elle dissimule une profonde révolution des méthodes, des architectures et des techniques de conduite de projet.

Le changement induit par le RAD est certes violent, mais moins brutal qu'une perte d'emploi. Je connais beaucoup de chômeurs informaticiens qui aimeraient être contraints à un projet RAD.

### 2.2 Principes fondamentaux

James Martin est le réel précurseur des méthodes de développement moderne. Dès la fin des années 80, il définit les bases du RAD. Pour qu'un projet puisse se qualifier de « RAD », plusieurs conditions doivent être remplies et plusieurs techniques doivent être mises en œuvre. Parmi l'ensemble de ces techniques, certaines ont plus de poids que d'autres, et il faut impérativement les respecter :

- la présence d'un **animateur spécialisé**. Il facilite les communications de groupe. Il doit obtenir un équilibre de participation entre les utilisateurs et les informaticiens. Il doit rendre compte à la direction générale et être rémunéré par elle, car sa neutralité est primordiale ;

- un **plan de communication** et de formation des utilisateurs. Il est planifié et approuvé sur la base d'un contrat fondamental au respect de la planification ;
- une organisation « plate » de **l'équipe (SWAT)**. Idéalement, les membres de cette équipe auront tous le même profil : concepteur-développeur, mais chacun disposera d'une spécialité complémentaire à celle des autres ;
- une **structuration du projet** en cinq phases distinctes produisant des livrables définis et le respect d'une dimension temporelle fixe « *The need for a 90 days life cycle* » (figure 2) ;
- un ensemble de **matériels et de logiciels performants**, isolés dans un environnement permanent, dédié en priorité au projet (salle RAD).

Le non-respect de ces points élémentaires dénie à un projet le qualificatif de RAD.

### 3.2.1 Les 5 phases du RAD

INITIALISATION (préparation de l'organisation et communication RAD)

Cette phase permet de définir le périmètre général du projet, de structurer le travail par thèmes, de sélectionner les acteurs pertinents et d'amorcer une dynamique de projet. Cette phase représente environ 6% du projet

CADRAGE (techniques RAD d'analyse et d'expression des besoins)

L'expression des besoins est effectuée par les utilisateurs qui présentent leur travail lors d'entretiens de groupe (session). Il est généralement prévu entre 2 et 5 jours de sessions par commission (thème). Cette phase représente environ 9% du projet.

DESIGN (techniques RAD de conception et de modélisation)

Les utilisateurs sont également impliqués dans cette étape. Ils participent à l'affinage et à la validation des modèles organisationnels : flux, traitements, données. Ils valident également le premier niveau de prototype présentant l'ergonomie et la cinématique générale de l'application. Il est prévu entre 4 et 8 jours de sessions par commission. Cette phase représente environ 23% du projet.

CONSTRUCTION (techniques RAD de réalisation, prototypage)

Durant cette phase, l'équipe RAD (SWAT), doit construire l'application module par module. L'utilisateur participe toujours activement aux spécifications détaillées et à la validation des prototypes. Plusieurs sessions itératives sont nécessaires. Cette phase représente environ 50% du projet.

FINALISATION

Des recettes partielles ayant été faites à l'étape précédente, il s'agit dans cette phase d'officialiser une livraison globale et de transférer le système en exploitation et maintenance. Cette phase représente environ 12% du projet.

## 2.3 Synthèse / phasage d'un projet moderne

Voici pratiquement et succinctement comment se déroule un projet RAD et les étapes de modélisation qui serviront de plan aux développeurs (figures 1 et 2).

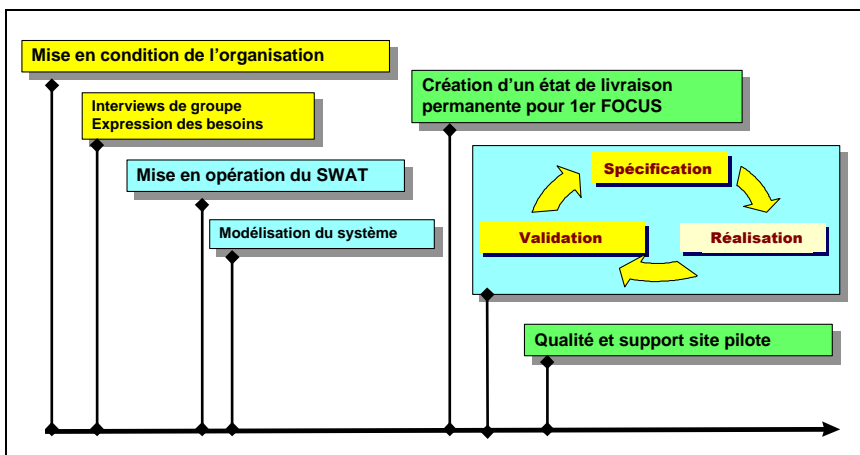


Figure 1. — Jalons décisifs du cycle projet RAD

**La première des conditions réside dans la présence d'un animateur ou d'un coordonnateur maîtrisant parfaitement tous les aspects du RAD.** Dans les projets où des dissensions pourraient apparaître entre les différents interlocuteurs ce spécialiste de l'entretien de groupe doit nécessairement être neutre.

La phase d'INITIALISATION prépare les intervenants aux contraintes d'un projet RAD. Après une courte IMMERSION dans le domaine fonctionnel, ils présentent à la maîtrise d'ouvrage, lors d'un ENTRETIEN PROPRIETAIRE, l'équivalent d'une étude d'opportunité, les contraintes de la méthode et un plan d'action à respecter. Un modèle de communication est produit. Si nécessaire, une opération de réingénierie des processus précède l'automatisation du domaine. Une RÉUNION DE LANCEMENT du projet est ensuite organisée. Elle regroupe tous les acteurs recensés et donne lieu à l'individualisation des travaux préparatoires à la modélisation du système

### Trois phases principales structurent un projet RAD, le Cadrage le Design et la Construction.

Le CADRAGE et la première partie du DESIGN, s'appuient sur une approche descendante (*top-down*) qui permet de respecter la cohérence systémique et les choix stratégiques ou opérationnels. La CONSTRUCTION privilégie la définition détaillée des besoins et applique une approche montante (*bottom-up*).

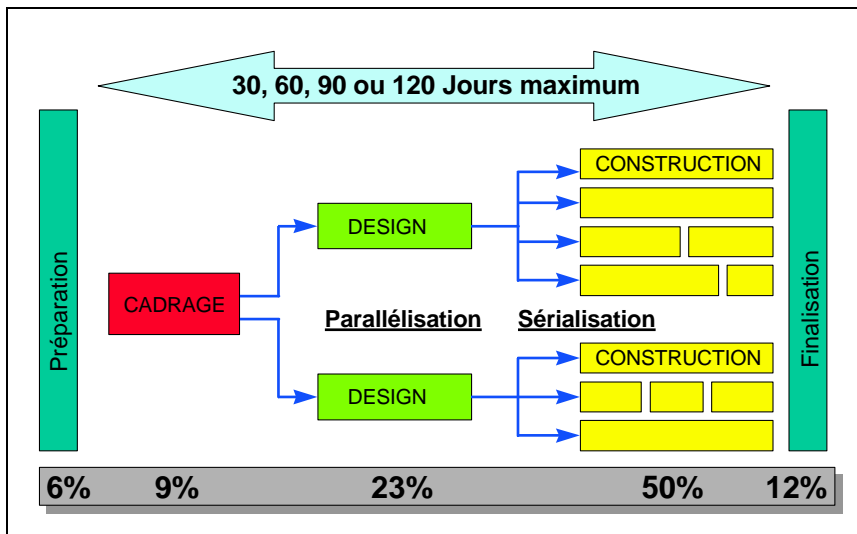


Figure 2. — Parallélisation et sérialisation des grands projets

Lors du CADRAGE auquel participent les décideurs, l'animateur obtient le verrouillage des besoins, des budgets, des délais et de la solution globale sur les plans stratégique, fonctionnel, technologique et organisationnel. Dans le cas où les besoins et les ressources divergeraient, une technique de type « Analyse de la Valeur » permet de d'attribuer des priorités aux fonctionnalités en termes de retour sur

investissement. Cette modélisation des traitements s'effectue sous la forme d'une hiérarchie de fonctions.

La phase suivante, le DESIGN, repose sur la disponibilité d'un AGL de conception léger et puissant. Cet outil est utilisé « en direct », dans une salle spécialement équipée de moyens de rétroprojection et de communication. Sous la coordination de l'animateur, les utilisateurs significatifs et les concepteurs-développeurs travaillent alors en commun et en direct à la modélisation détaillée des traitements et des données de l'application. La modélisation des données reste classique et s'appuie sur le modèle entité-relation. L'AGL de conception est capable de générer automatiquement et immédiatement la structure de la base de données à partir du modèle. La présentation d'un premier niveau de prototype conclut cette phase.

La phase de CONSTRUCTION affine le prototype. Elle fusionne les étapes classiques de :

- spécification détaillée ;
- réalisation (codage) ;
- tests unitaires ;
- tests d'intégration ;

ainsi que :

- la plus grande partie des tests de cheminement fonctionnel ;
- la recette initiale de l'application.

Cette homogénéité constitue, avec la prise de décision immédiate et la validation continue, les bases mêmes de la productivité et de la qualité RAD. Pour atteindre cette performance, les outils de Construction utilisés doivent être choisis avec soin (VB, Delphi, etc.), une charte graphique doit avoir été validée, des normes de programmation employées, un modèle de transaction généralisé à tous les modules et, si possible, le cadre d'une méta-application aura été respecté.

La phase de FINALISATION comprend, entre autres, la formation, la mise en œuvre d'un site pilote et les activités liées au déploiement général. Elle clôture le projet RAD.

Les projets majeurs impliquent une métrique précise et le respect d'une dimension temporelle des cycles. Il en découle un découpage en sessions parallèles. On optimise en planifiant des lots de 60, 90, 120 jours. Au-

delà, apparaissent la baisse de productivité, la démotivation et le turnover : un projet long est un mauvais projet.

A titre indicatif, il est utile de savoir, qu'en général le poids respectif des phases est le suivant :

- Préparation (6%) ;
- Cadrage (9%) ;
- Design (23%) ;
- Construction (50%) ;
- Recette RAD (12%).

L'engagement utilisateur minimum est de l'ordre de 12% à 15% de la charge incombant à l'informatique pour les applications de gestion classique.

Cette description n'est pas exhaustive, pourtant elle met en évidence que ceux qui considèrent le RAD comme une absence de méthode ne l'ont jamais pratiqué et ignorent tout du sujet.

Le RAD n'est pas le « chaos ». Le RAD consacre l'aboutissement d'une maîtrise organisationnelle, méthodologique et technologique.

Le RAD s'applique à toutes les tailles de projets. Il a pour objectif principal d'améliorer la qualité des développements, mais réduit les délais de production et facilite la maîtrise des coûts.

Le RAD ne remplace pas la modélisation du système par du prototypage mais utilise cette technique pour associer pleinement les utilisateurs dans une recherche de consensus et d'efficacité.

## 2.4 Votre projet est-il « RAD » ?

***Pour ceux qui désireraient vérifier si leur approche de la méthode ou celle de leurs prestataires respecte bien les règles d'efficacité du RAD voici les éléments d'une check-list succincte.***

### **Principes**

Pour maîtriser un processus qualité, il faut “dire ce que l'on fait, écrire ce que l'on dit et faire ce que l'on a écrit” ; aussi : votre approche méthodologique est-elle formalisée dans un document en détaillant, par étapes et par tâches, la mise en œuvre ?

### **Salle RAD**

Disposez-vous d'une salle spécialisée pour la communication de groupe, équipée de :

- matériels et de logiciels de bureautique ?
- moyens de rétroprojection électronique ?
- outils d'évaluation, de planification, de modélisation et de développement ?

### **Animateur**

Disposez-vous d'un spécialiste neutre, formé aux techniques d'entretien de groupe, maîtrisant les principes de modélisation du RAD ?

### **Formation**

Avez-vous fait dispenser aux utilisateurs significatifs et aux représentants de la maîtrise d'ouvrage une formation à la lecture des modèles (entité-relation, hiérarchie de fonctions) ?

### **Formation spécifique de la maîtrise d'œuvre**

Le personnel de la maîtrise d'œuvre est-il en mesure de remplir le rôle de rapporteur RAD (il synthétise en direct lors des entretiens le discours utilisateur et modélise en direct à l'aide d'un outil adapté et d'un rétroprojecteur) ?

### **Normalisation RAD**

Disposez-vous :

- d'un modèle de transaction standard ?
- d'une charte graphique acceptée ?
- de normes de codage ?

- de normes de documentation ?
- d'un dictionnaire d'abréviations et du vocabulaire « métier » ?

### **Respect des verrouillages**

Réalisez-vous toutes les étapes définies dans le plan et respectez-vous le verrouillage formel des cadrages et de la modélisation avant de procéder à la Construction ?

### **Disponibilité des utilisateurs**

Avez-vous fait signer par la Maîtrise d'Ouvrage un document contractuel engageant la disponibilité des utilisateurs ?

### **Coordination du projet**

Avez-vous réussi à créer des équipes autonomes et responsabilisées de concepteurs-développeurs travaillant en « coordination plate » ?

### **Outils (AGL)**

Disposez-vous d'outils pour produire et gérer simplement et automatiquement :

- un modèle de communication ?
- une hiérarchie de fonctions ?
- une évaluation de projet ?
- une planification ?
- un modèle de données ?
- un prototype ?
- une installation de logiciel ?
- une documentation intégrée ?

### **Poste de travail**

Avez-vous fourni aux développeurs un matériel performant (Pentium) doté d'un sous-ensemble vidéo de haute résolution (SVGA) et d'un écran en conséquence (17 pouces minimum) ? Les développeurs disposent-ils des outils de réalisation les plus puissants et les plus légers et de l'autonomie technique et financière suffisante pour acquérir sous 24 heures les « *add-in* » ou « *add-on* » indispensables à la qualité ou à la performance de leur production ?

### **Organisation**

Avez-vous fait étudier les documents et les circuits définis lors du Cadrage par un spécialiste en organisation (reengineering) avant d'engager les phases de Design et de Construction ?

Si vous avez répondu oui à l'ensemble de ces questions votre projet peut être qualifié de RAD. Dans le cas contraire cette simple évaluation sera certainement bénéfique à votre réflexion.

Le RAD se caractérise avant tout par un mode projet qui sublime l'organisation du développement en la plaçant hors des contraintes et des freins habituels. Les principaux prérequis cités le démontrent : **autant la motivation est forte pour ceux qui participent au projet, autant ce mode d'engagement est révélateur des faiblesses de l'organisation.** Il faudra donc réussir le projet dans un temps très court, sinon les tensions inévitables finiront par déstabiliser tous les intervenants. Le premier projet RAD doit être avant tout une démonstration qui engagera une réflexion plus profonde du reste de l'organisation. Avant ce point, à l'extérieur du projet, la pensée de beaucoup serait plutôt du type « pourvu qu'ils se plantent ».

## 2.5 Direction de projets stratégiques

Contrairement à ce que les théoriciens du « conceptuellement correct » avaient pensé, l'évolution des architectures techniques portée par le modèle client-serveur, la reconnaissance de Windows comme interface universel, la réduction des budgets et du cycle de vie des applications ainsi que l'implication organisationnelle de l'outil informatique ont totalement bouleversé le développement d'applications.

### 2.5.1 Réduction des budgets et du cycle de vie

Le chef de projet se trouve alors confronté à une mutation de son métier et parfois à une rupture profonde lorsque l'organisation qui l'emploie est importante et génère de l'entropie. **Réaliser des systèmes de complexité croissante, avec des produits parfois en cours de stabilisation, impose de la méthode, le sens du risque calculé et le goût de l'adaptation.** D'autre part, les développements informatiques évoluent d'un rôle d'accompagnement opérationnel (gestion, commande, suivi d'activité, facture, stock, etc.) à un rôle stratégique de production de services. On ne peut imaginer de nos jours une banque mettant en œuvre un nouveau type de prêts ou une compagnie d'assurances la couverture d'un nouveau risque sans que soit préalablement opérationnel le système d'information correspondant. L'informatique devient alors le support du produit, voire le service lui-même, considéré en tant qu'information manipulée.

Dans certains cas, la durée de vie des produits ne dépasse pas quelques mois, on parle même d'applications à la fois stratégiques et « Kleenex ». Il devient alors évident, que sans une évolution méthodologique, les

développements ne peuvent s'adapter à ce type de cycle de vie. En fait, l'enjeu d'un projet aujourd'hui est plus économique que technique : **il faut caler la démarche de production à la durée de vie du logiciel**. Pour permettre cette évolution, les technologies sont à la fois secondaires et primordiales. Secondaires, comme toute technique, mais primordiales car indispensables aux nouvelles applications qui représentent la voie de l'efficacité et de la différence concurrentielle. De plus, ces technologies ne sont pas neutres sur le plan organisationnel.

## 2.5.2 Dimension organisationnelle étendue

L'importance croissante prise par les exigences de clients avertis impose aux organisations une vision renouvelée de leurs structures.

La **vision hiérarchique** impose un découpage en domaines d'autorité communiquant par interfaces formalisés alors que l'**axe qualité** nécessite quant à lui un flux continu (workflow).

Dans l'entreprise à la recherche de performance, l'approche classique d'une hiérarchie appliquant des contrôles verticaux fait place à l'engagement de tous dans le processus horizontal de production. Les motivations théoriques des deux principes sont identiques, mais les moyens divergent pour une qualité de résultats sans commune mesure. Les applications « mutent », les équipes de projet se spécialisent, les principes du SWAT s'imposent.

## 2.6 Qualité et motivation des hommes

### 2.6.1 Les hommes du SWAT

L'ensemble du cycle de développement doit être réalisé par une équipe de base, totalement engagée dans son projet et en total accord avec la méthode qu'elle emploie.

La base de l'équipe RAD (SWAT) est un petit groupe d'informaticiens expérimentés qui doit avoir le sens de son identité. James Martin déclare : « *Ils doivent disposer de locaux spécialement aménagés et décorés, et la réalisation des engagements doit donner lieu à des célébrations, leur coordonnateur surveillera tout signe précurseur de burn-out.* ». Pour avoir travaillé dans des entreprises américaines, je n'ai rien contre ces principes. A Bell MC, le DG organisait des pique-niques à midi et fabriquait lui-même les hamburgers. Comme on dit là-bas, c'est « la cerise sur le gâteau ».

Utiliser des ressources d'un profil unique de type concepteurs-développeurs présente de multiples intérêts mais aussi des contraintes dans leur engagement.

Les ressources doivent intervenir simultanément dès le début réel du projet informatique.

Leur compréhension est alors globale et unique. Les utilisateurs ne sont pas sollicités à de nombreuses reprises. Lors de la modélisation détaillée, si le projet est conséquent, ces ressources se spécialisent dans un sous-projet pour **entrer « just in time » dans le flot du détail**. La formalisation et le transfert d'information sont minimisés ainsi que les risques d'erreurs inhérents à ces activités.

D. Vauquier, expert et auteur de nombreux ouvrages, a depuis plusieurs années compris l'avantage de l'homogénéité de cette double compétence. Dès le début des années 90, il a déclaré : « *La progressivité des développements conduit à un profil du nouveau développeur qui doit cumuler les compétences de conception et celles de programmation* » Il en est de même de B. Meyer : « *La conception utilise les mêmes mécanismes intellectuels que la programmation à un niveau d'abstraction plus élevé. On peut gagner beaucoup si l'on intègre les deux activités dans une démarche commune* »

Faire produire des dossiers détaillés par des analystes, puis les faire développer par des programmeurs laisse place à l'interprétation et à l'erreur. Cette étape est non seulement coûteuse, mais les documents produits sont inutilisables dans un développement moderne. A l'exception du descriptif du périmètre global, dans un projet informatique moderne, tout besoin détaillé doit s'exprimer sous la forme d'une modélisation (données, traitements ou flux). Sinon, il n'est pas représentatif d'une des étapes du développement.

La portée d'un texte et sa capacité d'autovalidation sont trop limités. Seule la modélisation exhaustive des données et des traitements peut garantir un niveau de compréhension concrètement matérialisable en application.

La composition d'une équipe de développement RAD requiert une attention toute particulière en ce qui concerne la dynamique des relations interpersonnelles. Une seule personne peut démotiver un groupe. Du côté des informaticiens, les compétences techniques, si elles sont nécessaires, ne sont pas suffisantes pour assurer des relations interpersonnelles harmonieuses. La maîtrise de Visual Basic ou d'un AGL n'est pas en soi la connaissance universelle. Des personnes exerçant

de grandes responsabilités ou dotées d'une culture technique ou organisationnelle exceptionnelle en ignorent totalement l'existence et l'emploi.

Il ne faut pas surdimensionner l'équipe de développeurs en termes de compétences ou de diplômes si le projet s'étale sur plus de six mois. Cette tendance permise par la conjoncture actuelle (chômage) conduit rapidement à l'insatisfaction et à la démotivation.

### 2.6.2 SWAT profil des ressources

Contrairement à des idées reçues, les programmeurs excessivement compétents et techniques sont à éviter. Tout au moins lorsqu'ils s'attachent à mettre en œuvre l'intégralité de leur astucieux savoir sans le partager. En effet, les langages de développement sous Windows disposent d'un nombre impressionnant de fonctions complexes. La diversité des solutions possibles est telle que la maintenance d'un module réalisé par un « top » développeur peut devenir presque inexploitable après son départ.

Les membres de l'équipe n'ayant pas l'expérience du RAD doivent bénéficier d'une formation théorique abordant les principes de conduite d'entretien, de modélisation directe et de prototypage actif.

**Une culture d'entraide est à la base de l'efficacité du SWAT.** Elle doit être initiée par le coordonnateur RAD, puis comprise et développée par tous les membres du SWAT.

#### **En pratique, il est indispensable de :**

- mesurer et récompenser individuellement l'échange de compétences et de services ;
- éliminer les craintes du « demandeur » ;
- stopper immédiatement les réactions négatives ;
- focaliser les récompenses globales sur les résultats d'équipe.

#### **Les avantages qui découlent de ces pratiques sont évidents :**

- suppression des temps de recherche ;
- élimination des travaux en double ;
- homogénéisation des procédures ;
- réduction de la taille de l'application.

Une équipe doit communiquer formellement dans un cadre déterminé, sinon, elle est inefficace et passe son temps à corriger ses propres erreurs. La mauvaise distribution des tâches, l'excès de parallélisme, la sous-estimation chronique du volume de travail ruinent les bonnes volontés individuelles. Il en est de même d'une utilisation dogmatique de la méthodologie. Les techniques de communications internes au SWAT sont détaillées en phase de Construction (ouvrage : Processus Qualité).

Il faut toujours s'attacher à rechercher le réalisme. On évite, par exemple, de tenter de faire « entrer » à tout prix la planification dans les délais en la manipulant sans vergogne. Confronté à un problème de ce genre, le RAD révisé à la baisse les fonctionnalités du projet, quitte à revenir sur cette décision si la réalité le permet ensuite, plutôt que d'accroître la taille du SWAT (figure 3).

La clé de la productivité est le développeur de haut niveau qui sait intégrer la conception et la réalisation. Son principal outil est le carnet de chèques qui permet l'acquisition des objets qu'il n'aura pas à réaliser lui-même.

L'expérience et la disponibilité d'une bibliothèque de produits complémentaires (*OCX*, *Add-in*, *Add-on*), de modules réutilisables et de transactions standards permettent une productivité plus de dix fois supérieure à celle d'un couple analyste et programmeur classique.

### 2.6.3 Ressources à temps partiel

**Dans un projet RAD, les membres du SWAT sont engagés à cent pour cent.** Les ressources ne disposant pas de cette disponibilité ont d'autres préoccupations que celle du projet. Elles peuvent participer au projet à divers titres, mais non comme membres du SWAT. Sur le plan informatique, ce profil correspond en général à des ressources internes engagées sur d'autres projets ou dans des activités d'exploitation. Leur expérience des aspects fonctionnels ou organisationnels sont des connaissances de grande utilité, aussi devront-ils être sollicités ponctuellement à titre d'experts du domaine concerné.

**Durant les phases de Cadrage et de Design, les ressources à temps partiel peuvent être déléguées au support de la maîtrise d'ouvrage pour faciliter l'expression des besoins et aux validations. Durant la Construction, il est souhaitable qu'elles assurent les vérifications qualitatives, fonctionnelles et techniques de l'application (AQ).**

Après la livraison, ces ressources disposent alors de connaissances permettant la maintenance applicative et corrective. Leur mission peut alors s'exprimer dans le cadre d'une équipe d'assistance technique clientèle (ATC). La notion de client ayant dans un contexte qualité le sens d'utilisateur du produit.

#### 2.6.4 L'esprit du SWAT

Au-delà du profil général et de la complémentarité spécifique des membres d'un SWAT, ce qui différencie avant tout une équipe de « gagnants », c'est l'esprit qui naturellement en émerge. Aussi, faut-il favoriser toutes les formes de participation extra professionnelles susceptibles de favoriser la cohésion de l'équipe. Une équipe de projet dont les membres n'éprouvent pas le besoin de déjeuner ensemble le plus souvent possible devient difficilement un SWAT. D'expérience, je peux assurer que l'humour en général, et la naissance d'un humour de « projet » en particulier, est un pas significatif vers la sensation d'appartenance qui caractérise les équipes « gagnantes ».

- ① L'effort requis pour réaliser un logiciel croît exponentiellement par rapport à sa taille.
- ② En réduisant de moitié les fonctions d'un logiciel de taille moyenne, vous réduirez à un tiers l'effort de développement.
- ③ Pour gagner 23% sur les délais, il faut augmenter le personnel de 75 %
- ④ L'accroissement de coût est alors de 33 %.

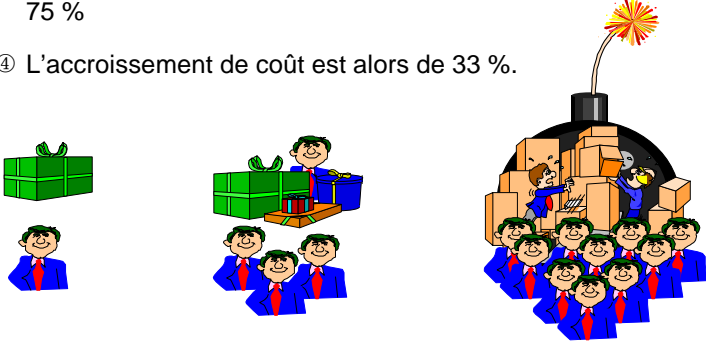


Figure 3. — Eléments de la problématique "taille du SWAT"

La plus grande partie des récompenses doit être collective, mais il faut récompenser individuellement toutes les actions de soutien afin de développer une vraie culture d'entraide. La performance a un prix, mais ce prix n'est rien en regard des économies qu'elle fait naître et surtout

des gains stratégiques qui en découlent. Tout est un rapport de retour sur investissement.

En 1982, je m'établis en Amérique du Nord, (c'est de ce continent que proviennent nos matériels et nos logiciels) et je compris que le Français est un individualiste tourmenté par la volonté de conceptualiser alors que le Nord-Américain dans sa naïveté se contente de réaliser ce qu'il croit possible, c'est-à-dire tout. « *Just do it* », telle est la devise.

Le RAD se consacre à l'essentiel. Il nous offre la possibilité d'atteindre cette puissance simplificatrice, du moins en ce qui concerne le développement d'applications. Alors essayons de simplifier et dans la limite de notre action prenons pour devise « *Just **RAD** it* ».

### 2.6.5 Communication externe au SWAT

**La communication du SWAT se base sur des publications ou des techniques comme la normalisation, la revue de projet, les jalons ZD et les Focus.**

La communication externe est considérée comme un facteur additionnel de réussite. Un organe de presse, même sommaire à l'en-tête du projet, est régulièrement publié. Pour un petit projet, une ou deux pages publiées une ou deux fois par mois, en fonction de la durée du projet, sont suffisantes. A travers ces quelques lignes, graphiques ou dessins (amusants de préférence), le projet prend vie, et les futurs utilisateurs se l'approprient. Surtout lorsque, nombreux, ils ne peuvent pas tous participer aux spécifications.

La publication régulière est aussi synonyme de bonne santé, c'est-à-dire d'avancement. Il faut d'emblée exclure toute langue de bois de ce type de communication. La plupart des gens n'en sont pas dupes, et très rapidement c'est toute l'autorité qui est sapée à travers le mensonge. Les sujets généraux doivent être traités sous un angle humoristique. Ainsi, sont rappelées les dates importantes du projet ainsi que les événements non moins importants de la vie des participants. L'arrivée de nouveaux membres ou de nouveaux matériels (attendus parfois depuis trop longtemps) peut ainsi faire l'objet d'une célébration spéciale.

**Tout est bon pour promouvoir l'intérêt et l'engagement. Une entreprise doit vivre au rythme de ses projets.** Lorsque l'on réussit, on peut tout se permettre, y compris de ne pas sembler trop sérieux. Néanmoins, la réussite doit toujours demeurer modeste, car elle nécessite en général la collaboration de toute l'organisation qui doit « supporter » le mode projet qu'elle implique.

Un rejet ou un frein du reste de l'organisation doit être évité à tout prix. Le meilleur moyen pour cela reste l'élargissement du partage de la satisfaction au plus grand nombre possible.

**Un système réellement efficace doit être « pro-actif », seule l'expertise permet cette efficacité. Il n'y a pas de miracle, la maîtrise de processus complexes est possible.**

Prenez pour exemple la construction d'un avion ou plus simplement d'un téléviseur ou d'un micro-ordinateur. Si les erreurs que font sans vergogne les gestionnaires actuels se retrouvaient dans ces produits, imaginez votre vie ! La prévision du déroulement d'un processus de production est une affaire de planification, d'ordonnancement et de contrôle. Tous ceux qui dirigent avec succès des projets plus ou moins complexes savent cela. Ce qui est évident, c'est que notre époque de modernisme et de complexité permet de moins en moins l'amateurisme.

Des outils existent, ils permettent d'accroître la productivité, de prévoir les effets induits des changements, de contrôler les processus complexes en évolution. Robots ou ordinateurs, les machines ne sont pas tout, il faut des humains pour les utiliser, des hommes compétents, des spécialistes dans le sens vrai du terme. En RAD, c'est cela l'esprit du SWAT : chaque élément conscient de sa place, de ses forces et faiblesses, de ses droits et devoirs.

**Participer à la création d'une équipe de fanatiques de la performance et de la réussite de projets est la plus belle chose qui puisse arriver à un développeur.**

La démocratie dans un SWAT s'exprime par une direction collégiale étendue à toute l'équipe. Lorsque une décision est prise, son exécution est immédiate, et, comme l'équipe entière a participé au choix, aucune remise en question personnelle n'est admise.

Tout détournement personnel de la décision doit être considéré comme une faute grave. Cela implique la constitution préalable d'une vraie équipe SWAT, c'est-à-dire composée d'informaticiens expérimentés maîtrisant les techniques du RAD. Si les membres de l'équipe ne maîtrisent pas ces techniques, une coordination réduite exclusivement aux membres expérimentés est préférable. Dans ce cas, les débutants (première participation à un projet RAD) assistent aux prises de décision sans pouvoir les influencer autrement que par la formulation de conseils (toujours bons à prendre). Les éléments les plus jeunes ont parfois des réponses techniques plus récentes et souvent plus adéquates.

Après plusieurs projets RAD, les choix généraux de développement sont acquis par tous, seules les décisions spécifiques au projet nécessitent une séance d'approbation. Dans un SWAT, la bonne foi et la logique président à chaque décision. Dans le cas où l'esprit du SWAT est dénaturé d'une manière ou d'une autre, il est préférable de demander sa mutation. Laisser la situation dégénérer se termine toujours mal, et il est ensuite trop tard pour dégager sa responsabilité.

## 2.7 Evolution du RAD et du reengineering

Les concepts du RAD ont été définis en 1991 par James Martin et adaptés à partir de 1994 par Jean-Pierre Vickoff :

- au contexte méthodologique français ;
- aux architectures Client/Serveur ;
- aux spécificités de l'interface Windows ;
- à l'évolution de l'intégration technologique.

Un processus qualité RAD est fourni gratuitement avec cet ouvrage. Les développements qui désirent se qualifier de RAD doivent le respecter. Ce processus répond aux exigences des 5 niveaux du CMM (Capability Maturity Model) ou modèle d'évolution des capacités de développement logiciel.

Le CMM est le standard en termes de mesure de la qualité et de la productivité des développements. Il définit et hiérarchise les pratiques de qualité du développement en niveaux qui couvrent :

- la reproductibilité du process ;
- sa définition ;
- sa maîtrise ;
- son optimisation.

## 2.8 Causes de rejets

Pour faire accepter les contraintes du RAD, il faut, en regard des chiffres et de l'inertie des professionnels, une prise de conscience de l'enjeu par les dirigeants. Au plus haut niveau des organisations, la direction doit comprendre que personne du côté informatique ne sort immédiatement gagnant d'un passage au RAD :

- Le chef de projet y perd en autorité. Il lui est nécessaire de s'impliquer comme membre à part entière d'une équipe qui se rétrécit.

- Les analystes et les programmeurs formés à l'ancienne rechignent à la reconversion. Ces deux métiers fusionnent dans un seul profil : le concepteur-développeur.
- Les sociétés de services redoutent la réduction des budgets et les exigences d'une performance accrue.
- Le directeur informatique est en première ligne, face aux réticences de la plupart des intervenants devant le changement qui les bousculent dans leurs habitudes.

Il faut admettre à la décharge de l'informaticien que les normes comme ISO, SPICE, CMM qui devraient le guider dans sa démarche de rationalisation sont à la limite de l'utilisable. Et que seuls des fanatiques de la normalisation se lanceront dans l'étude, l'analyse, la synthèse et l'adaptation de ces documents trop longs et trop abstraits.

**Mettre en œuvre le RAD requiert courage et motivation ou restriction budgétaire et obligation de résultat.**

Le processus doit être initié conjointement par les directions opérationnelles et la direction informatique sous l'égide d'un « facilitateur » neutre répondant à la direction générale.

## 2.9 Histoires d'aberrations

Ce que l'on constate malheureusement dans la plupart des projets actuels, c'est l'incompréhension grandissante des réelles contraintes d'une application informatique moderne. Les spécialistes qui devraient être des architectes tant au niveau de la conception que de la réalisation semblent désarmés par l'évolution de la typologie des projets modernes et des techniques de conception et de réalisation. C'est un peu comme si un entrepreneur ne savait pas lire ses plans et n'avait aucune idée de l'ordonnancement des travaux. Cette réflexion n'est malheureusement pas seulement caricaturale et dans certains cas la réalité est vraiment plus « grave » que ce que l'on peut imaginer, comme on dit au Québec.

J'ai participé un jour au lancement d'un projet majeur couvrant la totalité des fonctions d'une entreprise de services. Dès le début du projet et malgré mon opposition, une opération de formulation détaillée de l'existant sur papier fut imposée aux utilisateurs. Aucune approche de structuration ne fut mise en œuvre. Le tout fut présenté comme du RAD. Les utilisateurs s'exécutèrent pour le mieux et considérèrent avoir tout dit. Lorsqu'il fallut produire une structure utilisable pour des informaticiens à partir de ce fatras, vint le début des désillusions, et, dans le même temps, l'on continuait à détailler les domaines déjà

abordés alors que l'opération s'étendait à d'autres. Plus personne n'osait avouer l'inutilité du procédé et arrêter la machine emballée.

Considérant la métaphore de l'arbre, des feuilles étaient produites, mais il n'avait ni y arbre, ni tronc, ni branches pour les accueillir. Si on ramène cela au bâtiment, c'est un peu comme si l'on avait tenté de poser des tuiles sur une charpente inexistante. Ce que la physique interdit, l'informatique vous autorise à l'essayer. C'est le principe de la virtualité dévoyée et tout le drame des échecs informatiques. Car un choix erroné dans ce domaine est beaucoup moins visible et brutal qu'une tuile qui tombe de dix mètres.

## 2.10 Conditions d'échec

Dans une situation où l'animateur est supposé faire du RAD, mais où le pouvoir intermédiaire en place se refuse à accepter les contraintes de la méthode, la marge de manœuvre est très faible. Cela arrive en moyenne une fois sur deux. Si, au début d'un projet, devant le refus de formation et d'adaptation vous criez immédiatement « au feu ! » personne ne vous croit : vous êtes externe à l'organisation et personne ne voit ni feu ni fumée. Il est donc nécessaire de laisser la situation pourrir. Cela représente des coûts et une perte de temps, mais, à moins d'être suicidaire ou particulièrement engagé, il n'y a en général pas d'autre choix.

Facteurs déterminants	Retard	Dérive	Qualité
Spécifications sans les utilisateurs réels		X	X
Les concepteurs ne développent pas		X	X
Les FOCUS ne sont pas respectés	X		X
Des ressources sont à temps partiel	X		
Le reste de l'organisation ne s'implique pas en mode « projet »		X	
L'équipe projet ne maîtrise pas les outils	X		
Structuration et dimension temporelle ne sont pas respectées (120 jours maxi)	X		

Figure 4. — Facteurs d'échec recensés - 1

**Laisser le problème se développer ne signifie pas non-intervention. Au contraire, il faut régulièrement informer par écrit la personne en charge du projet des écarts à la méthode et des risques en découlant (figures 4 et 5).**

Il faut ensuite attendre le point où la situation est suffisamment dégradée pour qu'il soit évident que les moyens employés ne sont pas adéquats. Une analyse documentée de la situation suivie d'une synthèse doit alors démontrer point par point que le respect du RAD permet la résolution du problème. Une analyse plus critique peut alors s'appuyer sur les divers documents que vous aurez préalablement diffusés. Cette étape ne présente pourtant aucun intérêt à partir du moment où tous les intervenants, ayant compris le problème, sont prêts à jouer le jeu du RAD pour redresser la situation. **A ce point vous venez de clôturer une crise grave. Le projet s'oriente définitivement vers les techniques du RAD, ou alors il vous faut partir.** Le coup de barre doit être total. Il n'est plus question d'accepter les compromis, sinon, attendez vous à être considéré comme définitivement responsable de l'échec qui en résultera.

Facteurs déterminants	Déborde	Dérive	Qualité
L'équipe ne travaille pas en synergie	X	X	X
La modélisation est inadaptée	X	X	X
L'équipe projet n'est pas isolée	X		
La faisabilité technique est ignorée		X	X
Le matériel de conception et de développement est sous-dimensionné	X		X
Les outils ne sont pas homogènes		X	
Les outils ne sont pas universels	X		

Figure 5. — Facteurs d'échec recensés - 2 -

En résumé, un échec de projet ou une dérive est moins l'échec des hommes que l'échec de l'organisation qui n'a pas su les diriger et les contrôler. Le retard de projet ou la non-conformité du projet est donc imputable avant tout à l'organisation. C'est l'échec des dirigeants à travers leurs actions ou leurs inactions.

Lorsque, et c'est souvent le cas, les hommes sont changés sans qu'aucun correctif ne soit apporté à l'organisation, il y a peu chances que la deuxième tentative soit plus satisfaisante, bien qu'elle puisse en général s'appuyer sur les leçons de la première.

## 2.11 Quels projets pour le RAD ?

Certaines questions reviennent régulièrement en conférence. Elles portent souvent sur la typologie de projet la plus propice au RAD. En fait, il n'y a pas de projet type pour le RAD. Il y a juste des organisations plus ou moins aptes à accepter les contraintes du RAD, pour ensuite pouvoir en apprécier les bénéfices. Il est possible d'appliquer le RAD sur un projet de dix ou quinze jours, comme sur un projet de trente années ou plus. Néanmoins, au-dessous d'un certain seuil de quelques dizaines de jours, seule une organisation disposant déjà de l'expertise et de l'environnement de travail RAD peut obtenir des bénéfices immédiats avec cette méthode. **Le premier projet n'est pas en général le plus économique, compte tenu des investissements de tous ordres nécessaires pour créer un milieu propice à la performance RAD.** Pour conclure comme M. de La Palice, c'est de l'incapacité de l'organisation à appliquer le RAD que découle l'impossibilité de réaliser un projet en RAD.

Prenons pour exemple pratique et réel une étude préalable. Elle démontrait qu'un projet classique évalué à 290 mois-homme pouvait être réalisé en 140 mois-homme si l'on adoptait une démarche RAD :

- La planification classique démarrait avec deux analystes. Après six mois, huit analystes-programmeurs devaient intervenir et l'équipe devait encore augmenter par la suite pour atteindre douze personnes durant la phase de réalisation.
- La planification RAD se basait sur l'engagement immédiat et constant d'une équipe RAD de cinq concepteurs-développeurs.

Comme aucun budget de projet n'avait été officiellement alloué, il était plus facile de commencer avec la solution classique. Et il me fut dit : puisque nous n'avons pas les moyens de faire du RAD, il faut employer la méthode classique. Je pleure avec vous, gestionnaires.

## 2.12 Les nouvelles applications

Depuis de nombreuses années, l'informatique n'a été pour le chef d'entreprise qu'un outil d'aide à la gestion. L'ordinateur était une super calculatrice qui représentait une ligne douloureuse de son bilan, mais généralement un faible pourcentage de son chiffre d'affaires. Depuis, insatiable, constamment évolutive, l'informatique s'affirme comme un

outil stratégique incontournable qui englobe progressivement toutes les fonctions de l'entreprise. Gestion, production, commercialisation, communication, organisation, aucun secteur n'y échappe, tous les processus en deviennent totalement dépendants.

Le chiffre initial représentant quelques pour cent du chiffre d'affaires peut vite devenir le poste le plus important du compte d'exploitation. Le cas est fréquent dans les entreprises de services. Aussi est-il vital de comprendre l'impact de cette transformation sur la structure même des projets informatiques. Les causes profondes des échecs de systèmes, dont le pourcentage explose depuis quelques années, sont alors directement mises en évidence, alors qu'une étude basée sur la seule nature des projets est tout a fait insuffisante à l'analyse des facteurs déterminants. Nous pouvons citer, parmi les éléments dont l'accumulation rend les systèmes instables ou trop complexes à réaliser :

- l'évolution des architectures ouvertes et du client-serveur ;
- l'apparition de technologies nouvelles non stabilisées (Workflow, Internet) ;
- la sophistication liée à l'interface graphique ;
- la mutation des outils de conception et de développement ;
- l'accroissement de la complexité applicative ;
- la réduction de la durée de vie des composants ;
- la multiplication des intervenants et des décideurs dans les applications transdomaines ;
- la nécessité de supporter un flux transversal orienté qualité et « client » ;
- les changements organisationnels et l'application de nouveaux styles de management.

Et, pour conclure cette liste, il faut inclure la mutation des méthodologies, contraintes de composer avec l'ensemble de ces paramètres. Tous ces points ont altéré profondément la nature des projets mais beaucoup d'informaticiens n'en sont pas encore concrètement conscients. Les nombreux échecs de projets auxquels nous assistons actuellement sont la conséquence directe de cet état de fait. Une formation permanente et hautement technique aura dans les années à venir une place prépondérante à jouer. De nombreux gestionnaires de l'informatique devront retourner « aux études », puis se replonger dans le concret de la technologie.

## 2.13 Organisation existante et reengineering

**Le reste de l'organisation doit adopter le mode projet dans ses relations avec l'équipe RAD. Ce qui implique un appui évident de la direction générale.**

Dans la recherche de l'efficacité tout est bon. Je me suis vu acheter sur l'heure, avec une carte de crédit personnelle, un modem, une dizaine de logiciels et un écran de 21 pouces. Une autre fois, ce fut des ordinateurs dans un supermarché. Ces acquisitions permirent à des ressources facturées l'équivalent de 3500 francs par jour d'être immédiatement rentables sur leurs projets.

Si l'utilisateur de base ne parvient pas à se dégager des contingences de son travail, il faut louer les services d'un intérimaire lorsque cela est fonctionnellement possible. De plus, il sera interviewé avant son départ, car toutes les visions neuves et les informations externes sont intéressantes à considérer.

Aucun frein technique ou organisationnel ne doit empêcher la bonne marche du projet. La réussite technique et économique est à ce prix. Les diverses actions que je viens de décrire pourraient sembler du luxe ou de l'extravagance. Elles ont été en fait à la source des plus grandes économies et réussites de projets qu'il m'a été donné de constater. D'ailleurs, si les divers gaspillages liés aux temps d'attente étaient chiffrés, bien des dirigeants en resteraient effarés et imposeraient ensuite le mode projet dans bien d'autres domaines.

Voici une autre anecdote mettant en œuvre ce principe et qui fera certainement rêver plus d'un chef de projet français. Au Canada, à Bell MC, je devais choisir un logiciel de fax. Cette fonction de communication devait être automatisée au cœur d'une application stratégique. Le projet respectait une dimension temporelle de 30 jours. En début d'après-midi je me rendis dans le « Surcouf » local (plus vaste et moins pénible) et achetai les cinq meilleurs produits, toujours avec une carte de crédit personnelle.

De retour au bureau, je les testai en recherchant particulièrement la facilité d'intercommunication avec Visual Basic. L'étude me prit environ trois heures et, le soir, à 17 heures, avant de partir j'annonçai au directeur : ce sera WinFax Pro, que dois-je faire des autres ? Mettez-les aux archives, répondit-il, et n'oubliez pas de vous faire rembourser par la secrétaire ! Ce directeur était tout à fait conscient des économies de tous ordres qu'il venait de réaliser tout en bénéficiant du choix technologique le plus judicieux.

## 2.14 Puissance et simplicité du RAD

Pour appréhender concrètement la puissance simplificatrice du RAD, il faut caricaturer le principe d'une simple tâche que l'on aurait à accomplir.

Dans le cas d'une méthode classique :

- Une réunion est organisée, elle a pour but de préciser ce que l'on souhaite faire.
- Les personnes disponibles, intéressées ou curieuses participent, mais les décisionnaires préfèrent attendre la synthèse de la réunion.
- Dans le meilleur des cas, un des participants se voit confier la mission de formaliser ce qui a été décidé et d'expliquer comment il est certainement possible le réaliser.
- Une nouvelle communication fait part aux intervenants de ce qui devrait être fait.
- La recherche d'une personne-ressource responsable d'un résultat est lancée.
- Un malchanceux semblant disponible est appréhendé.
- Les informations réunies sur le sujet lui sont communiquées.
- L'insuffisance de consensus nécessite de réunir encore une fois les intervenants disponibles pour explications renouvelées et décisions que l'on espère définitives.
- La boucle est renouvelée jusqu'à acceptation de tous ou abandon de certains.

La réalisation se déroule suivant un processus similaire et aboutit généralement à une crise alors que le travail semblait achevé.

**Dans une session RAD, sont réunis simultanément suivant un plan de communication préalablement accepté, ceux qui :**

- savent quoi faire,
- savent comment faire,
- décident de faire.

Pour des raisons d'efficacité, dans le cas de l'absence d'une des composantes, la session est reportée. Une fois l'information transmise et l'accord obtenu, l'application se réalise avec les conseils et sous le contrôle du « client ».

## 2.15 La méthode Merise et le RAD

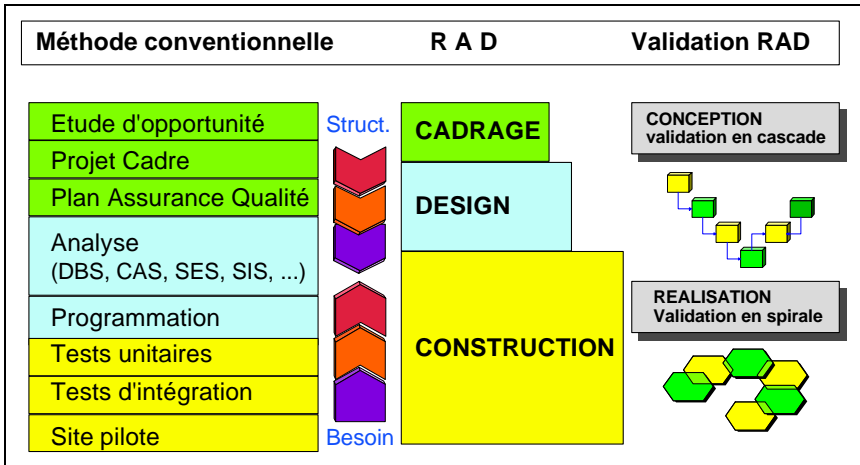


Figure 6. — Comparatif des cycles de développement

**L'évaluation du RAD, par rapport au CMM, met en évidence la réalité et la profondeur des techniques mises en œuvre par cette méthode.** La différence entre Merise et le RAD est structurelle (*figure 6*). Le phasage est totalement différent. La dimension temporelle, omniprésente en RAD, n'existe pas dans Merise. En ce qui concerne la modélisation, RAD utilise des techniques et des **raccourcis** à travers les niveaux d'abstraction qui sont totalement inavouables pour un « Merisien » convaincu. Il en est de même de l'intégration des outils dans la méthode ainsi que des techniques de communication. En fait, le RAD est un ensemble de « techniques sensibles » à mettre simultanément en œuvre avec beaucoup de finesse. Ne pas le comprendre peut conduire à l'échec d'un projet aussi sûrement que l'emploi dogmatique de Merise. Aussi faut-il se méfier plus que jamais de ceux qui commencent à vendre du conseil RAD sans avoir l'expérience concrète de nombreux projets de typologie et de taille différentes.

Le RAD à l'opposé de Merise ou de SDM/S recouvre une méthodologie et les bases d'une méthode totalement adaptable à la complexité croissante des développements modernes et aux nouvelles contraintes économiques. Au-delà des désillusions engendrées par la recette « Merise assaisonnée de RAD », il faut pour réussir un projet RAD mettre en œuvre la méthode dans son intégralité. D'ailleurs de façon générale, il faut toujours choisir une méthode et ensuite s'y tenir. Dans ce domaine, les compromis sous couvert de souplesse sont en réalité la cause des principales dérives.

**Le RAD n'est pas une technique complémentaire ou l'aboutissement de Merise. C'est au contraire une approche « collision », complètement perpendiculaire à la pensée Merisienne.**

**Il est terrifiant de constater le nombre de personnes qui apposent aujourd'hui le label RAD sur des méthodes qui n'en sont pas.**

## 2.16 Difficulté de l'animation de projet

**L'animateur RAD est un crucifié permanent** (figure 7). Son rôle est presque toujours mal compris. Souvent il n'intervient que pour la communication, l'organisation générale et pour l'animation des sessions de travail. Pourtant, on lui demande fréquemment de tout connaître du projet et même parfois d'en savoir plus que les ressources engagées à plein temps. Il doit maîtriser parfaitement les nouvelles architectures, l'intégration technologique et les techniques de modélisation. En fait, en plus des connaissances inhérentes à sa fonction, on exige de lui les compétences techniques de la maîtrise d'œuvre et les compétences fonctionnelles de la maîtrise d'ouvrage. A la moindre erreur, c'est lui qui paie. D'autre part, lorsqu'il veut rester neutre, comme rien n'est plus subjectif que la notion de neutralité, il peut s'attendre au pire. Citons René pour illustrer ce propos « Depuis toujours les justes meurent mutilés pour s'être exposés nus au toucher du bien »

C'est dans ces conditions que j'ai vu se « faire virer » un certain nombre d'animateurs brûlés par quelques jours de sacerdoce. La palme dans ce domaine revenant certainement à la société cliente X. Elle venait d'abandonner un projet de plusieurs centaines d'années-homme et pour repartir du bon pied, il fut décidé de réaliser un premier projet RAD. Un cycle de 90 jours avec quatre développeurs semblait être une bonne mesure de test. Un animateur de la compagnie de services la plus prestigieuse et la plus expérimentée dans ce domaine fut engagé. Après deux ou trois réunions on le remercia : il était trop brutal dans sa recherche d'efficacité. Avait-il remis révolutionnairement en question l'organisation ? non : il avait jeté un stylo sur la table pour faire cesser des bavardages intempestifs. La raison donnée pour le sacrifice du second fut son manque de partialité. Lorsque l'on sait que le projet était en fait un développement interne, on reste songeur. Que lui serait-il arrivé dans le cas d'un forfait traité en externe ? Dans les faits, l'informatique n'avait pas l'habitude d'un arbitrage et d'un interlocuteur sachant évaluer les problèmes réels liés à chaque demande.

Un animateur doit dépendre de la direction générale, sinon autant se passer d'animateur, cela évitera bien des hypocrisies.

A cet égard, je veux évoquer une entreprise où le service informatique soutenait à la direction générale s'être converti aux pratiques du RAD. Il paya pendant plusieurs mois un animateur sans pour autant accepter de mettre en œuvre une seule des techniques de cette méthode. L'otage avait un choix : abandonner son contrat ou se taire.

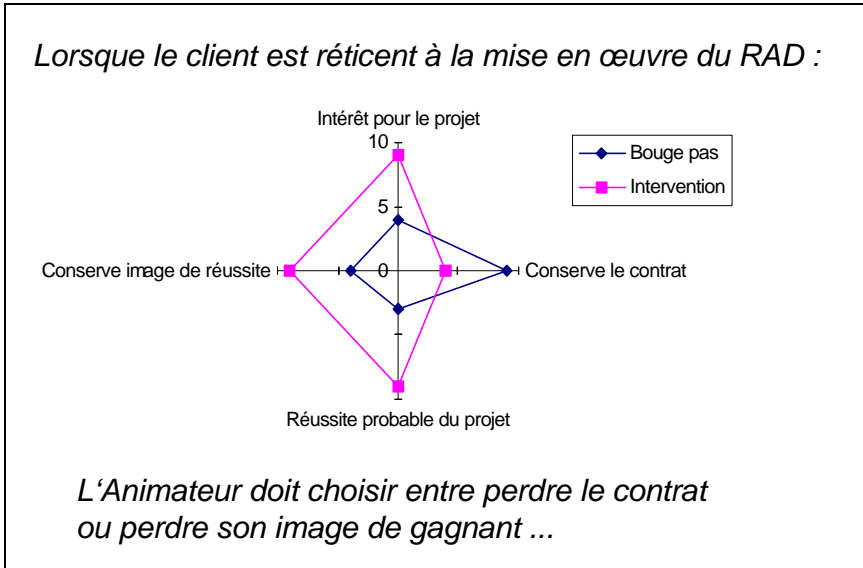


Figure 7. — Dilemme classique de l'animateur RAD

## 2.17 Culture française et reengineering

Il n'est pas étonnant que le RAD, synonyme d'adaptation, ait pour origine l'Amérique du Nord et qu'il ait tant de mal à s'imposer en France. La culture organisationnelle des cadres français est basée sur la compétition et non sur la collaboration. En Amérique du Nord, depuis l'université, à travers la formation et le sport, la notion d'équipe prédomine. En France, les élites préfèrent les sports individuels. Cela fait partie des raisons pour lesquelles le *groupware* a autant de mal à être mis en œuvre. Il est aussi intéressant de constater qu'il débouche sur des abus incroyables de réunionnisme lorsque la technologie facilite trop des prises de rendez-vous, imposées automatiquement et simultanément à de nombreux intervenants. L'outil devient alors un exutoire à une libération des communications informelles ou parallèles trop longtemps encadrées et censurées.

Il n'est pas un projet où les problèmes de direction hiérarchisée ne prennent le pas sur le bon sens. Quelle que soit la taille de l'équipe, il faut que l'un des membres dirige les autres. Parfois les conditions de

nomination dépassent l'entendement et le sens commun. La contrepartie de ces errements est toujours la même : désintérêt et improductivité.

**Contrairement à ce que certains pensent, la direction collégiale est possible. Il suffit simplement qu'un noyau de participants connaisse les règles du jeu et soit motivé. En RAD cela se nomme un SWAT, une équipe de projet.**

En France, de nombreuses personnes de formation élitiste sont totalement incapables de travailler en équipe. Dès leur arrivée dans un simple projet de développement, elles s'affairent à mettre en œuvre des circuits parallèles d'information et des accords de partage de pouvoir. Au Canada, ces manœuvres existent, mais elles sont en général discrètes et réservées au niveau supérieur de la hiérarchie des très grandes organisations. C'est ce que l'on appelle « politique d'entreprise ».

Dans les projets informatiques, les dissensions sont rares. L'intérêt de la société est primordial. La réussite du projet est considérée comme un succès d'équipe. Chaque individu se sent concerné par la tâche qui lui est confiée, et personne ne prendrait le risque d'être déconsidéré aux yeux de ses collègues et patrons pour des raisons aussi futiles.

Depuis mon retour en France la plupart des problèmes que j'observe sont l'aboutissement d'un manque de coordination acceptée ou d'une absence de directives clairement exprimées. Des projets stratégiques sont passés bien près de la catastrophe pour ces raisons issues d'un autre âge du management.

## 2.18 Les pompiers du reengineering

En Amérique du Nord, la consultation informatique consiste le plus souvent à réaliser une application « clé en main ». L'expérience et les réussites précédentes assurent généralement la coordination du projet. Jusqu'au début des années 90, les buts étaient clairs, le consultant était le plus souvent considéré comme une main-d'œuvre additionnelle ou spécialisée dans un cadre de développement précis. A l'heure actuelle, le recruteur ne cherche plus une ressource, il espère une réponse. Le plus souvent, cette réponse ne lui convient pas, car elle part du postulat que tout change. Il lui faut remettre en question ses habitudes, ses certitudes et parfois même s'interroger sur les fondements de sa culture.

Pour avoir exercé dans de nombreux pays, je sais combien il est difficile de communiquer, de diriger ou d'être dirigé et plus généralement d'accepter l'« autre ». Il n'en est pas moins étonnant que ce soit en France, dans mon pays d'origine, que je rencontre le plus de difficultés à

initier le changement. D'ailleurs, je ne dirige plus de projet, je me contente de les orienter autant que faire se peut vers le succès. Le niveau de résultat dépend du bon vouloir et de l'état d'esprit de cadres plus ou moins sensibles à un changement qu'ils n'ont pas ressenti ou qu'ils préfèrent ignorer.

**Il est aussi de plus en plus rare que me soit confiée une réelle responsabilité dans une application à son début. Mon rôle est devenu celui d'un « pompier » chargé de limiter l'impact d'un échec engagé.**

Les causes de l'échec sont généralement flagrantes. Le plus souvent on constate la présence de ressources n'ayant aucune raison d'être engagée dans un projet informatique moyen. Des personnes dotées d'un très haut niveau d'études et surtout d'individualisme, tentent de diriger leurs pairs, qui le ressentent très mal. Les problèmes sont en général dissimulés sous un couvert feutré de pseudo politique d'entreprise. Au bout du chemin, quelques mois plus tard, l'échec, puis le coup de gomme et l'oubli. Pas de *mea culpa* ni d'expérience capitalisée, mais le plus souvent un nouveau départ sur des bases identiques. La capacité d'auto-régénération du cadre informatique est exemplaire et aurait pu inspirer Darwin. Il n'est donc pas étonnant que les outils, normes ou indicateurs capables de jauger et de juger la productivité soient totalement ignorées par la plupart d'entre eux.

Mais, plus sérieusement, devons-nous réellement cantonner ces ingénieurs issus des plus prestigieuses écoles dans de petites tâches de réalisation ? La récession justifie-t-elle ce gâchis ? Oui, une crise qui engendre le chômage et la surqualification est bien présente. Mais ne serait-il pas plus efficace de redéployer ces compétences dans des travaux stratégiques à la dimension de leur valeur ? Energie et motivation seraient alors certainement au rendez-vous d'une nouvelle expansion.

## 2.19 Evolution et causes de rejet

**De nombreux cadres et dirigeants n'ont toujours pas compris que la survie même de leur entreprise passait par une évolution.**

Lors d'un projet récent dans une entreprise autrefois publique, j'ai dû faire face à une réaction d'une violence étonnante. Le personnel n'avait pas compris l'ampleur des changements que nécessitait l'adaptation de son organisation dans un secteur devenu hautement concurrentiel.

Au cours d'une réunion à laquelle participaient des cadres opérationnels, j'avais relaté comment en Amérique du Nord, les gens travaillaient,

réagissaient aux changements et ce qu'ils offraient comme services aux clients. La comparaison portait sur la permanence dans l'effort de qualité et surtout détaillait les techniques organisationnelles nécessaires à la compétitivité. A la fin de la conversation, je sentis bien un froid, mais je n'avais vraiment pas compris à quel point ces changements les rebutaient. Quelques jours plus tard, il me fut interdit d'aborder de nouveau ce sujet. La bêtise l'avait emporté sur la curiosité, car ces propos dérangent leur confort intellectuel. Ils ne voyaient pas ce qu'ils avaient à gagner à imaginer ce genre d'avenir, fût-il le présent pour d'autres et une source de solutions pour leur futur.

Cette réaction me surprit réellement. Généralement, à travers le RAD, l'informaticien passe pour un « héros » aux yeux des utilisateurs. De nombreuses années de consultation au Québec m'ont par ailleurs appris à soigner, dans la modestie, mes premiers contacts. Ce qui aboutissait en général, après quelques semaines, à une déclaration du genre : « Finalement, il n'est pas pire, ce Français », première étape d'une acceptation méfiante.

## 2.20 Le besoin du client (interne ou externe)

La finalité de l'organisation est de concevoir, de développer et de livrer des produits ou services à ses clients. **Le flux « guide » de la qualité (figure 8) traverse latéralement les différentes fonctions de l'entreprise. Les secteurs d'influence verticaux ne le concernent pas.**

Parmi les divers processus, celui qui relève du service client est de loin le plus important. Dans cette perspective, le BPR conduit à revoir l'organisation en partant du besoin client. C'est concrètement une rupture par rapport aux principes d'organisation actuels. Ce changement de vision nécessite de gérer les ressources matérielles et humaines selon des contraintes différentes. Cette rupture est si difficile à mettre en œuvre que certains experts, comme James Martin, préconisent, plutôt que de tenter d'adapter des organisations réticentes, d'en créer de nouvelles hautement productives, puis de laisser mourir les anciennes. La nouvelle structure basée sur une organisation et un système d'information entièrement modélisé et évolutif s'appelle une "Cybercorp" (organisation cybernétique).

Au Canada, un bon exemple est Bell Mobilité Cellulaire. Cette entité indépendante a été créée de toute pièce à côté de la lourde Bell (téléphonie classique) qui n'aurait pas su résister à la concurrence. Cette organisation est en permanence confrontée aux initiatives de ses concurrents. Son salut réside dans sa réactivité à travers une organisation et un personnel toujours en mouvement, prêts à se former, à évoluer et à

s'adapter. C'est au niveau de l'organisation "apprenante" que le souci du partage du savoir (forums, projets qualité) entraîne une fertilisation croisée des compétences. Les salariés développent de manière autonome des initiatives qui servent à la mise en œuvre de la stratégie de l'entreprise grâce à des applications rapidement réalisées et déployées

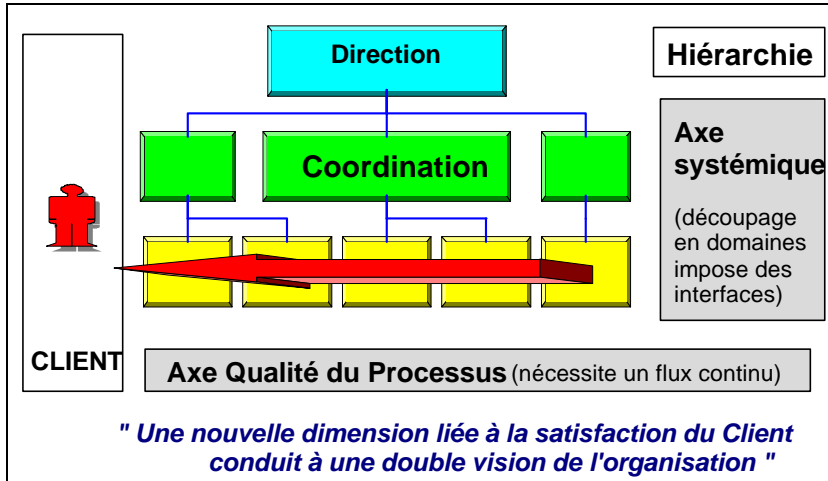


Figure 8. — Dimension organisationnelle du flux qualité

**Seules des applications légères issues de la bureautique communicante ou des principes du RAD sont en mesure de répondre à un tel besoin de réactivité.**

## 2.21 Une vision globale du développement

La méthodologie RAD organise une vision globale du développement d'applications et en englobe tous les composants :

- la conduite de projet (logistique et pilotage) ;
- la gestion des communications interpersonnelles ;
- la composition et la coordination des équipes ;
- les méthodes de conception ;
- l'outillage logiciel ;
- les techniques de réalisation ;
- l'intégration technologique.

Cette envergure nécessite en premier lieu des ressources informatiques cohérentes avec les besoins de développements informatiques de l'entreprise.

Contrairement aux idées répandues (on ne sait sur quelle base), le RAD recommande d'ancrer les développements fondamentaux sur un socle solide issue d'une planification globale de type plan directeur. Les premières étapes du RAD, y compris la phase d'analyse (Cadrage), sont d'ailleurs de type systémique (*top-down*). Au niveau de la conception (Design), on s'attache immédiatement à définir dans l'essentiel, une structuration des données qui offre ensuite le moindre effort de développement.

Le RAD est aussi affaire d'organisation, de timing et d'optimisation de problèmes humains. A juste titre, Marcel Soberman précise : « *Un développement de type RAD repose non seulement sur une méthode et des outils, mais aussi sur une organisation liant utilisateurs et informaticiens dans des sessions intensives, dans des échanges fréquents, et sur un rythme d'activité permettant la délivrance de produits dans des temps suffisamment brefs pour que les équipes ne se dissolvent pas, le contexte reste stable, les besoins à l'origine du projet n'évoluent pas trop.* » Cela recouvre en fait toute la problématique du développement d'applications modernes, et malheureusement très peu de responsables informatiques semblent en être conscients.

Dans un projet RAD, l'implication des utilisateurs est primordiale. La maîtrise d'ouvrage est responsabilisée et engagée dans un rôle opérationnel en regard du projet. Cette réelle responsabilité de production implique une formation préalable, car les utilisateurs classiques ne sont généralement pas familiers de ce genre de participation.

**Le RAD impose une réactivité maximale aux intervenants et fixe un cadre structurant. L'organisation se trouve contrainte à l'efficacité.**

## 2.22 Histoires d'architecture technique

Dans le cycle d'abstraction de Merise, l'aspect physique se situe au dernier niveau. Ce qui consacre la prédominance du conceptuel sur le matériel. La plupart des autres méthodes de conduite de projet, dans une optique identique, considèrent les contraintes liées au hardware que tardivement dans le processus. Les projets modernes ont bouleversé cette vue.

**L'intégration technologique se situe désormais au cœur et autour de chaque projet client-serveur. Certains projets n'ont d'ailleurs d'existence que par les nouveaux matériels qu'ils mettent en œuvre.**

L'informatique nomade en est un exemple. De même, la gestion des ventes ou des stocks à partir de l'étiquette intelligente utilisant les

technologies de « radio fréquence ». Dans un projet moderne, le coordonnateur technique est avant tout un excellent intégrateur de technologie. Il dispose du pouvoir et des moyens d'acquérir les outils qui lui sont nécessaires. Aujourd'hui, on peut dire (et redire) que le carnet de chèques est le principal outil du développeur.

Pour en revenir à l'architecture technique proprement dite, ce qui caractérise 90% des projets de gestion est leur similitude en termes de puissance de traitement (CPU / mémoire) et l'accroissement des communications réseau.

En généralisant, on peut présenter comme plate-forme universelle le Pentium sous Windows NT. C'est tout aussi réel pour les serveurs multiprocesseurs ou en clusters, que pour les stations de travail. Seule, la mise en évidence de particularités extrêmes nécessite de consacrer plus de temps à une étude technique approfondie.

## 2.23 Histoires d'outils non appropriés

Dans les précédentes éditions de mes ouvrages sur le RAD, un chapitre entier était consacré à l'axe technologique. Un avertissement préalable limitait la pérennité de cette information. Dans le présent ouvrage, cette partie n'est pas actualisée. Il reste pourtant essentiel de considérer le RAD comme une approche méthodologie « outillée ». Ce qui va sans dire nécessite d'être souvent répété.

**Les outils de conception et de développement les plus conviviaux et les plus légers sont paradoxalement les plus puissants et les moins coûteux.** En mécanique, que ferait le professionnel avec une pince comme seul outil de base ? Il est vrai que si l'on tient un marteau bien en main, tout ressemble à un clou. La finesse des interfaces modernes et la simplicité que requiert leur mise en œuvre efficace nécessitent une boîte à outils complète. La notion de robustesse, critère fondamental voici encore quelques années, ne signifie plus rien aujourd'hui. A ce sujet, il me revient à l'esprit l'expérience d'une entreprise qui abordait son premier projet en Visual Basic. En réservant mes services, le responsable du projet, m'avait fait analyser les besoins d'un tel développement. Ce gestionnaire efficace savait que les diverses formalités de la commande prendraient plusieurs semaines. Je lui proposais une liste minimum de quatre OCX complémentaires qui permettaient de constituer un environnement de développement professionnel. Le montant de la commande devait être de l'ordre de cinq mille francs.

Deux mois plus tard, les concepteurs-développeurs arrivaient comme prévu, mais le processus d'acquisition était bloqué. Un responsable des

choix techniques ne comprenait pas, n'en voyait pas la nécessité, pensait que c'était « dangereux ». Mieux, il déconseillait d'utiliser les produits de développement Microsoft. Il avait réalisé une étude, pour justifier un produit concurrent, dont la synthèse s'exprimait sous la forme sophistiquée d'un graphe de type « radar ». Derrière l'apparente neutralité de l'étude se dissimulaient des considérations très partiales. Seuls les paramètres techniques étaient pris en considération. La pérennité de la solution Microsoft, sa totale homogénéité (VBA, OLE) ainsi que les points décisifs qui positionnent Microsoft comme le leader incontesté de l'informatique actuelle avaient été ignorés. Cette personne, très compétente sous d'autres aspects, n'appartenait pas à une équipe de développement. Elle ne répondait pas à des contraintes de productivité telles que celles qui sont requises d'un SWAT. Dans ses choix, la sophistication technique l'emportait toujours. La courbe réelle d'apprentissage imposée par les produits n'était pas prise en compte. Il n'y avait aucun terrain d'entente entre ces principes et le RAD. Le principe d'autonomie du SWAT était bafoués, sa productivité s'en trouvait pénalisée.

Pour l'expert RAD, la priorité numéro 1 du choix d'outils est toujours le rapport entre la facilité d'utilisation immédiate et la puissance de développement standard.

Lors du choix de l'outil de documentation en ligne, cette déviance imposa un produit particulièrement lourd et ne disposant pas d'une version française. Je préfère oublier les divers problèmes et pertes de temps qui découlèrent naturellement de ce détail. Pour le reste, il fallut plus d'un mois, sur un projet RAD planifié à soixante jours, pour obtenir finalement la moitié des outils indispensables prévus initialement. Durant ce projet, il ne se passa pas une semaine sans que l'aspect technique qui aurait dû nous apporter des solutions ne nous pose des problèmes. Le choix de la résolution minimum de développement en regard de la taille des écrans fut un autre exemple d'incompréhension et de gaspillage de ressources qui justifia une annexe dans mon premier livre RAD.

## 2.24 Méthode, bon sens et reengineering

Comme il a été préalablement détaillé, les statistiques démontrent un accroissement récent des échecs en développement de projets informatiques. Parmi les causes principales, on relève le manque de communication mais aussi l'accroissement de la complexité des besoins « utilisateurs » et le décalage entre les techniques de modélisation et cette complexité.

Tout ces projets avaient néanmoins des points communs :

- ils étaient dirigés par un chef de projet confirmé ;
- ce chef de projet avait bénéficié d'une excellente formation ;
- ce chef de projet mettait en œuvre une méthode éprouvée ;
- pendant la plus grande partie du projet, ce chef de projet se considérait sur la bonne voie ;
- de nombreux documents démontraient la rigueur de l'approche ;

et finalement l'application concrète n'était pas au rendez-vous.

Pourtant, en prenant un peu de hauteur, ces chefs de projet auraient pu se rendre compte de la situation et prendre des mesures simples pour redresser la barre. **Tout cela découlait plus du bon sens que de la maîtrise de techniques complexes.** Généralement, les membres de l'équipe étaient conscients des problèmes, et « radio café » en parlait tous les jours. Il faut croire que, dans une tête bien formée, il y avait un trou noir à l'endroit où aurait dû se situer le sens commun.

## 2.25 Les raccourcis méthodologiques

Paradoxalement, c'est aussi bien l'absence de méthode que son excès qui aboutit aux échecs. La plupart des débuts de catastrophe sur lesquels il m'a été donné d'intervenir avaient pour point commun la poursuite chimérique d'une perfection sur « papier » des livrables. Ces travaux semblaient indispensables au respect de la méthode alors que s'égrenait inexorablement la réserve de temps et d'argent allouée au projet.

Il ne faut pas confondre la méthode et sa raison d'être. L'aboutissement doit être une application opérationnelle et non un stock de papier justificatif d'un effort, aussi louable soit-il.

**Actuellement, il est peu probable de réaliser un projet sous contraintes sans emprunter des raccourcis méthodologiques.**

Prendre un raccourci est le plus souvent synonyme de prendre un risque pour en réduire un autre. Pour qu'une prise de risque soit payante, il faut quelle soit calculée, donc que le nombre d'inconnues soit raisonnablement limité. Le plus classique est la réduction du nombre de niveaux d'abstraction, il en existe de nombreux autres mais il faudrait un livre pour les décrire et les sécuriser. Un raccourci n'est possible que lorsque la typologie du projet le permet, et sa contrepartie impose une plus forte implication de l'utilisateur dans la réduction du risque qu'il induit.

Structurer pour réduire la complexité est aussi le meilleur moyen d'obtenir la puissance fonctionnelle en optimisant les coûts de conception et de réalisation. Il faudrait un livre entier pour tenter d'expliquer comment modéliser en pratique. L'architecture de la conception est donc primordiale. On ne le répétera jamais assez, un bon architecte maîtrise et met en œuvre des pratiques de conception comme l'exhaustivité globale, l'abstraction, la modularité, l'encapsulation, la généralisation, etc.

## 2.26 Histoire de conception inadaptée

Pour comprendre le danger réel que fait courir une conception inadaptée aux développements modernes, voici l'étude d'un cas réel que nous nommerons le projet X de l'entreprise X.

L'entreprise X disposait d'une grande expérience des développements classiques et bénéficiait des services d'une cellule spécialisée en méthode (Merise, SDM/S). Le projet X se composait d'une application nomade de saisie d'informations, d'un sous-ensemble communication et de deux applications centralisées de gestion et d'exploitation des données.

L'équipe prévue pour ce projet stratégique se composait d'un directeur de projet (grande école) et de 4 chefs de projet. Un des chefs de projet avait la responsabilité de l'architecture technique et des communications. Les trois autres chefs de projet « fonctionnels » étaient chargés de réaliser la conception générale et la conception détaillée.

Durant la programmation, chaque chef de projet « fonctionnel » devait voir son équipe s'étoffer de trois ou quatre développeurs. A son point d'orgue ce projet allait donc compter plus de quinze personnes.

L'entreprise X engagea des chefs de projet et débuta la phase d'expression des besoins. Le but était de respecter le cycle merisien de la « courbe du soleil ». La conduite du projet suivait la méthode SDM/S. Les premières étapes consistaient en une modélisation organisationnelle de l'existant. Elles devaient aboutir à une modélisation conceptuelle de l'existant, puis orienter la modélisation conceptuelle du nouveau système. Ensuite, étaient prévues les étapes de modélisation organisationnelle et finalement de modélisation logique qui devaient aboutir à la description des programmes et de leurs fonctionnalités. C'est à partir de ces dossiers que les développeurs pourraient programmer.

Le raisonnement était sans faille, c'était intellectuellement aussi beau qu'une symphonie. La cellule méthode comptait observer l'esthétisme des passages de relais.

La date butoir était fixée à fin décembre de l'année suivante (18 mois). Chacune des cinq premières étapes devait prendre environ deux mois, quatre mois étaient prévus pour la réalisation, deux mois pour les tests et trois mois pour le site pilote. Avec un petit effort, cela devait entrer.

Dans la réalité, l'outil de base disponible pour produire les livrables de ces étapes, n'était pas un AGL, mais ABC Flow charter (c'est mieux que PaintBrush). Les difficultés qui résultèrent de ce choix aboutirent à l'utilisation d'Oracle Designer sous une émulation Unix. Les postes de travail étaient sous-dimensionnés, à base d'écrans de 14 pouces. Il était impossible de visualiser plus de quatre entités simultanément. Il est possible de résumer ces conditions par un seul mot : l'Enfer. Afin d'avoir une vue d'ensemble, les flux étaient dessinés sur les murs. Cette initiative, haute en couleur, me valut, de la part de la secrétaire, le surnom de « Picasso ». Il me resta jusqu'à la fin du projet.

La conception commença, les utilisateurs étaient disponibles et participèrent. Les premiers retards de formalisation furent mis sur le compte des vacances. Après cinq mois, on gribouillait toujours de « l'organisationnel existant ». Le sixième mois fut une période d'abattement psychologique. Le découragement suintait des modèles, la révolte grondait. Elle s'exprima au bout du neuvième mois.

Le directeur de projet n'était pas engagé concrètement et techniquement dans la production des livrables inutiles qu'il imposait.

A ce point, la moitié du temps de développement avait été gaspillé. En coulisse (radio café), la charge était estimée au double de ce qui était officiellement prévu. Sur le plan modélisation, c'était le néant. Seules des informations générales existaient sur papier. Très intéressantes pour produire un rapport, elles s'avéraient inutilisables dans le cadre de la précision requise pour l'indispensable modélisation des données et des traitements.

Il devenait évident que les chefs de projet étaient individuellement incapables de produire des dossiers pour quatre programmeurs. Alors que la méthode aurait (déraisonnablement) impliqué de doubler leur nombre. Il était tout aussi clair que le nombre d'analystes et les contraintes de l'interface graphique rendaient impossible un tel exercice de formalisation sur papier.

La compagnie changea alors de statut, le budget développement fut réduit de moitié et l'ensemble des intervenants fut prié de donner dans la productivité. Les propriétaires de places confortables furent promus

locataires de sièges éjectables et s'intéressèrent donc de plus près à l'avancement du projet. Le « super » directeur du projet et le chef de projet technique furent écartés.

A ce point, un marché fut proposé aux trois chefs de projet survivants. Il leur fallait terminer eux-mêmes, dans le délai initial et avec le budget restant, ou se démettre. Heureusement l'un d'eux avait l'expérience d'une méthode de développement rapide. La cellule méthode, consternée, se retira, considérant que le projet courrait désormais à sa perte.

Deux des chefs de projet savaient développer et acceptèrent la réalisation. Le troisième reprit la direction du projet et s'occupa plus particulièrement de l'administration des données. Une secrétaire fut allouée pour faciliter le minimum de formalisation à maintenir. Une ressource additionnelle, excellent développeur, vint compléter l'équipe de conception-réalisation. Elle se lança alors dans la modélisation des données suivie d'un prototypage actif.

**Pour garantir un déploiement complet, sécurisé par une exploitation en site pilote d'au moins trois mois, il fut décidé de donner des priorités aux fonctionnalités et de livrer en trois lots successifs.** Les utilisateurs étaient sollicités plusieurs fois par semaine pour participer à la création des prototypes. **Les participants se rendirent alors compte de l'inutilité des documents initialement et chèrement produits, qui furent totalement abandonnés.**

Deux mois plus tard, des présentations de l'application en cours de développement étaient réalisées (toutes les deux ou trois semaines). Ces Focus mobilisaient environ une vingtaine de personnes qui critiquaient et enrichissaient l'application, fonctionnalité par fonctionnalité.

**Une spécialiste en documentation réalisa alors la documentation utilisateur.**

Une première version de l'application comprenant environ 60 % des fonctionnalités initiales fut stabilisée en six mois. Le mois suivant fut consacré au test d'intégration et de communication. L'exploitation du site pilote démarra sans trop de difficultés. Le déploiement s'effectua normalement trois mois plus tard. La cellule méthode ne comprit pas comment nous avons pu réussir et mit cela sur le compte du hasard (qui fait bien les choses). Les utilisateurs et les informaticiens furent récompensés par deux jours de fiesta. *Happy ending!*

Il faut tirer plusieurs leçons de ce projet :

- Premièrement, il fait apparaître qu'une méthode hautement sécurisante et structurée conduit à l'échec ;
- Deuxièmement, il démontre qu'un petit nombre de ressources performantes et motivées (bien payées, entre autres) peut sauver une situation où une équipe nombreuse aurait échoué par enlèvement.

**Pour une cellule méthode, l'information principale à tirer d'une telle expérience se situe dans les techniques de modélisation à employer.**

On ne peut concevoir, modéliser, gérer, une application sous Windows en architecture client-serveur comme une application classique. Si l'application couvre plusieurs domaines ou mélange de l'opérationnel et du décisionnel, ces spécificités doivent être prises en compte. Elles nécessitent d'utiliser des techniques de modélisation adaptées à chaque cas :

- Dans une partie de l'application, la difficulté vient de la modélisation des données ;
- Dans une autre partie, ce sont les flux qui sont difficiles à cerner ;
- Ailleurs, la complexité des traitements s'avère le problème majeur. Dans tous les cas, les formes de modélisation doivent être adaptées ainsi que les techniques de conduite de projet.

# Pour une performance indispensable

*« Pour réussir, il ne suffit pas de prévoir. Il faut aussi savoir improviser. »*

*Isaac Asimov (1920-1992), Fondation*

### 3. Pour une performance indispensable

Dans les pays industrialisés, la mondialisation est synonyme de délocalisation de la production. Les industriels justifient ces opérations par le différentiel de coût salarial dont ils bénéficient dans des pays du tiers-monde. C'est donc par la suppression de centaines de milliers d'emplois que se réalise chaque année la désertification industrielle.

La seule réponse possible reste la performance. Au cœur de ce défi se trouve l'outil informatique dont l'usage éclairé doit nous permettre de résister encore quelque temps en accroissant notre niveau de productivité et de qualité. La recherche de l'adéquation entre offre et besoins nous mènera à l'ultime personnalisation des produits que seule l'automatisation parfaite rend possible (*high-tech & high-touch*).

Comme on le comprend, l'informatique est au cœur de l'évolution et la conditionne. L'organisation essaie de répondre aux attaques en accroissant sa compétitivité, mais l'inertie et le conservatisme de sa structure freinent tout effort visant à l'améliorer comme à la déstabiliser. Cet état réactionnaire ne serait pas mortel dans un monde fermé aux communications et aux échanges. Malheureusement, l'ordinateur sur votre bureau est d'origine américaine, votre indispensable Windows est américain, Internet est américain.

Dans la recherche d'une réponse adaptée, certains pensent que miraculeusement le RAD permettrait d'aller plus vite sans remise en cause de l'organisation des développements. Il n'en est rien. La lecture de l'ouvrage original de James Martin démontre au contraire que le RAD est avant tout la mise en œuvre d'une organisation parfaite des développements. Depuis, il est devenu tout aussi évident que la réussite des nouveaux projets implique simultanément la réorganisation de la fonction informatique et celle de la fonction à informatiser. Ce constat n'est d'ailleurs pas le fait du RAD. Il découle d'une vision pragmatique de la qualité de service intimement liée à la compétence des cadres opérationnels. C'est dans cet esprit qu'il faut lire l'annexe « Qualité : pensez globalement » où le concept est abordé sous la forme critique d'un constat d'absence de « voie royale ».

Sur le plan des méthodes, la différence entre ce qui est réellement appliqué dans les entreprises et ce que l'on voudrait nous faire passer pour un standard est affolante. Dans le pays de Descartes, où Merise, l'approche la plus structurée que l'on puisse concevoir est en œuvre depuis vingt ans, l'absence d'intérêt pour l'organisation des développements est flagrante. Pour s'en convaincre, il suffit de visiter le

forum « méthode » du *Monde Informatique*. Il représente sans ironie un louable effort et le plus grand forum ayant jamais été offert gratuitement aux informaticiens confrontés à la méthode et à la conduite de projets. Malgré des présentations régulières durant 1996, moins de dix informaticiens sont intervenus. Quatre des rédacteurs étaient des coutumiers de la publication, et la plupart des autres messages se limitaient à quelques lignes de questions. Voilà où nous en sommes fin 1997. La cause de cette absence de participation est-elle un manque d'intérêt pour le sujet ? Une incapacité à utiliser Internet ? Le rejet de la gratuité qu'impose ce genre de forum ?

Cela a amené un des participants, Ph. Hamant, professeur à l'AFPA, à la réflexion suivante : *«Ce forum manque singulièrement de punch et d'activité compte tenu de son importance. Comment faites-vous, les concepteurs de ce temps ? N'avez-vous pas de méthode ? Comment feront les autres ? Où sont sur ce forum les constructeurs et les distributeurs des méthodes de demain ? »*

En attendant la perfection des méthodes de demain, utilisons le RAD et ses raccourcis. Mais surtout comprenons bien qu'ils ne signifient pas une absence de méthode. L'aboutissement de ces deux visions (absence et raccourcis) est totalement opposé malgré les apparences. Le résultat pratique est soit la réussite d'un projet sous contraintes, soit l'échec par l'épuisement des ressources.

Dans cette recherche d'optimisation, la plupart des informaticiens surestiment leurs connaissances de la conduite de projet et de la modélisation. Le raccourci méthodologique nécessite :

- la compréhension des formes variées de modélisation ;
- la finesse dans le choix des techniques et des outils ;
- le sens de l'expression du détail utile ;
- la mesure des risques liés aux raccourcis.

Dans de nombreux domaines de l'informatique de gestion, la recherche de la perfection est une dangereuse chimère. La recherche d'une qualité optimale dans un contexte donné en fonction de contraintes précises est en fait le seul but pouvant être atteint. Cet abandon de la notion de perfection est le principe le plus dur à accepter intellectuellement pour l'esprit logique d'un informaticien. C'est pourtant la recherche de l'absolu qui amène des projets à se perdre « corps et biens ».

Refuser la demande d'un utilisateur n'est pas chose aisée. Il est préférable, pour éviter un traumatisme, d'obtenir un renoncement de sa part. Pour rendre cette situation possible, il faut chiffrer le coût de

chaque fonctionnalité. L'enveloppe budgétaire doit être confiée à l'utilisateur. Celui-ci prend alors conscience de la somme dont il dispose et du coût de ses besoins. De lui-même, il s'impose des priorités. Si cela est possible et vraiment indispensable, il propose un budget additionnel. Souvent cela signifie aussi de négocier les délais. La gestion du budget en regard des fonctionnalités doit être le dilemme du client (l'utilisateur), pas celui de l'informaticien.

Dans les applications stratégiques ou complexes, ce qui souvent va de pair, seules ces techniques de *design to cost* permettent de maîtriser l'explosion de la demande. Les techniques de *design to cost* et les raccourcis méthodologiques sont les deux piliers de la réussite d'un projet RAD. Le pilotage par les directions opérationnelles, à partir du budget et des délais, est la seule forme de conduite de projet qui puisse faire face à la complexité des développements modernes.

Un des autres dilemmes de l'informaticien se situe en regard de l'organisation qu'il doit informatiser. Bien souvent la formalisation des flux, lorsqu'elle est effectuée pour la première fois, fait apparaître des dysfonctionnements. Dans de nombreuses entreprises nord-américaines, chaque projet informatique était précédé d'une étude des processus qui permettait d'adapter le système à la nécessité de nouvelles conditions de travail. Dans une entreprise en évolution permanente cela se concrétisait par un léger « tuning » et non par une réorganisation majeure.

En France, cette pratique salvatrice est rare. Pourtant, à travers la notion de DIO (direction informatique et organisation), il avait été compris quel rôle la fonction informatique devait jouer parallèlement à la fonction organisation. Récemment, dans une grande entreprise française, des directeurs opérationnels auxquels venaient d'être présentées les améliorations organisationnelles préalables à un développement d'application, s'exclamèrent « *on n'a pas demandé à ces informaticiens de réorganiser la compagnie !* ». Je comprends humainement cette réaction, mais il est difficile à un homme de logique de découvrir des dysfonctionnements, de les laisser perdurer, et de développer une application pour les automatiser.

Claude Salzman, vice-président de l'AFAI (Association française de l'audit et du commerce électronique), dans un article intitulé *Peut-on tout dire au client* expliquait : « *L'expérience prouve que la seule vérité est celle que le client veut entendre.* » et ajoutait « *Il y a une résistance au bon sens. Des équipes qui ont perdu le sens commun ne sont plus capables d'entendre le discours de la raison.* », puis il reprenait « *Dans ce type de situation, tout le monde sait bien que cela va mal ... Mais ces personnes sont tenues par des solidarités...* ». Cet article se concluait par une distinction entre la tâche de

L'auditeur axée sur la norme et la conformité, et celle du consultant axée sur la recherche de l'optimisation des moyens et des résultats.

En regard de mon expérience nord-américaine, je pense réellement que la France doit développer une culture dédramatisée de la réingénierie des processus. Cette pratique doit nécessairement accompagner une organisation en mouvement. Autant cela semblait évident au Canada, autant en France je n'ai jamais vu mettre en œuvre cette approche en douceur. Au contraire, on essaie de masquer jusqu'à la fin du projet les aspects organisationnels qui pourraient inquiéter, en espérant que l'application agisse comme un révélateur après coup. Bien souvent la prise de conscience est trop tardive et l'on développe inutilement une application inadaptée qui pérennise le problème de fond. Il y a un long chemin à parcourir avant d'arriver à un système d'information optimal dans la plupart des organisations. Pour cela il faudra comprendre et admettre la réelle nécessité de la critique d'un œil extérieur penché sur chaque projet.

L'homme est faillible. Aussi, des processus formalisés lui sont-ils indispensables à la maîtrise de son action. Il lui faut ensuite une farouche volonté ou obligation de les respecter et l'intelligence de les adapter. De ces contraintes naît la capacité de piloter dans la qualité et la performance les activités les plus complexes. C'est au prix de cet « état de l'art » et à celui de la simplification que le professionnel obtiendra finalement l'amélioration continue d'une qualité qui visiblement et statistiquement lui fait défaut. Soyons positifs, si 83% des projets s'achèvent dans l'échec, donc 17% aboutissent. Il faut impérativement se donner les moyens d'être à 100% dans ces 17%.

Récapitulons la recette : une ingénierie d'organisation comme le CMM, une méthodologie de conduite de projet telle que le RAD, des méthodes et des techniques de conception-réalisation adaptées à chaque projet. Mais évidemment rien ne saurait être aussi simple, il faut donc en plus quelques ingrédients : le savoir-faire et l'engagement des hommes, car, comme dit le diable de la *Divine Comédie*, « tout est dans la finesse ».

## 4. Annexes : CMM & UML

### 4.1 Le CMM en résumé

CMM: Capability Maturity Model ou modèle d'évaluation et d'évolution des capacités de développement logiciel a été mis au point par le Software Engineering Institute. Comme SPICE (ISO), CMM répertorie des pratiques de planification, d'ingénierie et de gestion qui, lorsqu'elles sont appliquées, améliorent la capacité de l'organisation à atteindre des objectifs de coûts, de délais, de qualité et de fonctionnalités. Cinq niveaux précis permettent à l'organisation de se situer et de fixer son objectif d'amélioration.

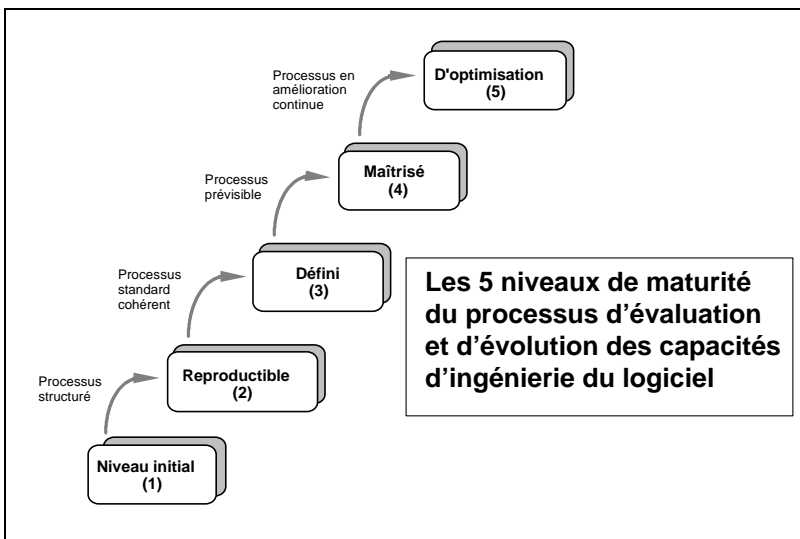


Figure 9. — CMM : 5 paliers dans la rigueur

#### 4.1.1 Niveau 1 "initial"

L'organisation ne dispose pas de procédures formalisées de développement et d'évolution de ses applications. Lorsque l'échec de projet se matérialise, le peu de méthode est généralement abandonné pour tenter des raccourcis dans le processus de réalisation et de validation. Les efforts d'organisation régressent vers des pratiques d'engagements purement réactives de type «codage et tests» qui amplifient la dérive. *Le RAD n'a aucun point commun avec ce niveau.*

#### 4.1.2 Niveau 2 "reproductible"

La gestion des nouveaux projets est fondée sur l'expérience mémorisée à l'occasion de projets semblables. *En RAD, cela correspond au respect de la*

*structuration des phases, de la dimension temporelle, de la validation des livrables, des architectures de conception et de réalisation. La planification s'effectue en phase de Cadrage à partir de la hiérarchie des fonctions à produire. La gestion des nouveaux projets est fondée sur l'expérience mémorisée à l'occasion de projets semblables. Le RAD rencontre ces exigences avec l'usage d'outils comme l'Évaluateur.*

#### 4.1.3 Niveau 3 "défini"

Le processus standard de développement et d'évolution logiciel est documenté. Il intègre en un tout cohérent les procédés d'ingénierie logiciel et de gestion de projet. *En RAD c'est le processus qualité d'ingénierie des développements.* Un programme de formation est en place dans l'organisation afin que les utilisateurs et les informaticiens acquièrent les connaissances et les compétences nécessaires pour assumer les rôles qui leur ont été confiés. *L'animateur RAD impose cette formation en préalable au début du projet et de préférence avant la réunion de lancement.*

#### 4.1.4 Niveau 4 "maîtrisé"

L'organisation se fixe des objectifs quantitatifs et qualitatifs. La productivité et la qualité sont évaluées par le contrôle et la validation des jalons majeurs du projet dans le cadre d'un programme organisationnel de mesure. *En RAD ces contrôles et ces mesures s'effectuent lors de la mise en place des Jalons « zéro-défaut » et des revues de projet dont l'aboutissement est « l'état de livraison permanente » qui permet la réalisation de « Focus » de validation et d'avancement.*

#### 4.1.5 Niveau 5 "optimisé"

L'amélioration continue des processus est la principale préoccupation. L'organisation se donne les moyens d'identifier et de mesurer les faiblesses de ses processus et renforce ceux-ci de façon proactive afin de prévenir les dysfonctionnements. Une cellule de veille technologique identifie, acquiert et met en œuvre les produits innovants et les meilleures pratiques d'ingénierie logiciel dont la synergie permet l'amélioration continue de la qualité. *Le RAD offre naturellement les moyens de répondre à cette dernière exigence par l'engagement de tous dans le cadre d'équipes d'amélioration de la qualité (interview de groupe, validation permanente). Pour les informaticiens, cette notion d'équipe se matérialise dans le SWAT (spécialisation, veille technologique, autonomie, réactivité, recherche de la performance et de la qualité au moindre coût). Pour la maîtrise d'ouvrage, ce sont les techniques d'enquêtes de satisfaction, de Benchmarking, et de veille concurrentielle).*

## 4.2 UML : utilité pour un gestionnaire

UML (langage unifié de modélisation) permet la description des éléments d'un développement logiciel, mais n'impose pas de processus de développement en tant que tel. UML est donc avant tout un langage dont l'utilisation peut déborder le strict cadre de l'informatique pour faciliter la modélisation de toute forme de processus. Au-delà de la simple notation UML, propose une organisation des différentes informations qu'il permet de décrire.

Actuellement, des comptables et des gestionnaires sans connaissances informatiques utilisent la notation UML, dans la limite des « use case », pour modéliser les circuits administratifs.

La technique des « use case » popularisée par Ivar Jacobson est un moyen pratique et particulièrement accessible aux utilisateurs désireux de formaliser leur activité. Pour être efficaces, les « use case » doivent respecter une unicité de thème, de lieu ou de temps. Les activités ne répondant pas à ces critères font l'objet de traitements indépendants, à l'instar des actes d'une pièce de théâtre utilisés pour marquer les ruptures de contextes ou les transitions temporelles.

Sans être totalement exhaustifs dans le détail, les scénarios doivent être représentatifs de toutes les situations à considérer. Au-delà d'une aide intéressante à la validation des modèles, les « use case » définissent les cheminements fonctionnels et sont particulièrement utiles aux informaticiens. Ils facilitent les validations lors des sessions de travail, des Focus et lors de la recette générale avant déploiement. Cette dernière étape représente souvent une tâche ardue dans les grands projets. La technique des « use case » se révèle alors un investissement intéressant.

**La technique des « use case » est un complément utile pour réduire le risque dans des situations fonctionnellement complexes.**

Exemple : La *figure 10* représente le niveau 1 d'un diagramme de flux utilisant la notation UML. La *figure 11* est un sous-ensemble de niveau 2 décomposant la fonction « P1 ». Ce diagramme n'est pas un cas d'école réalisé par des étudiants comme on en trouve dans la plupart des ouvrages. C'est une partie extraite d'un processus réel complexe. Il a été produit par des contrôleurs de gestion (Dumazedier@Expertises.com), après une heure ou deux de formation et quelques tâtonnements.

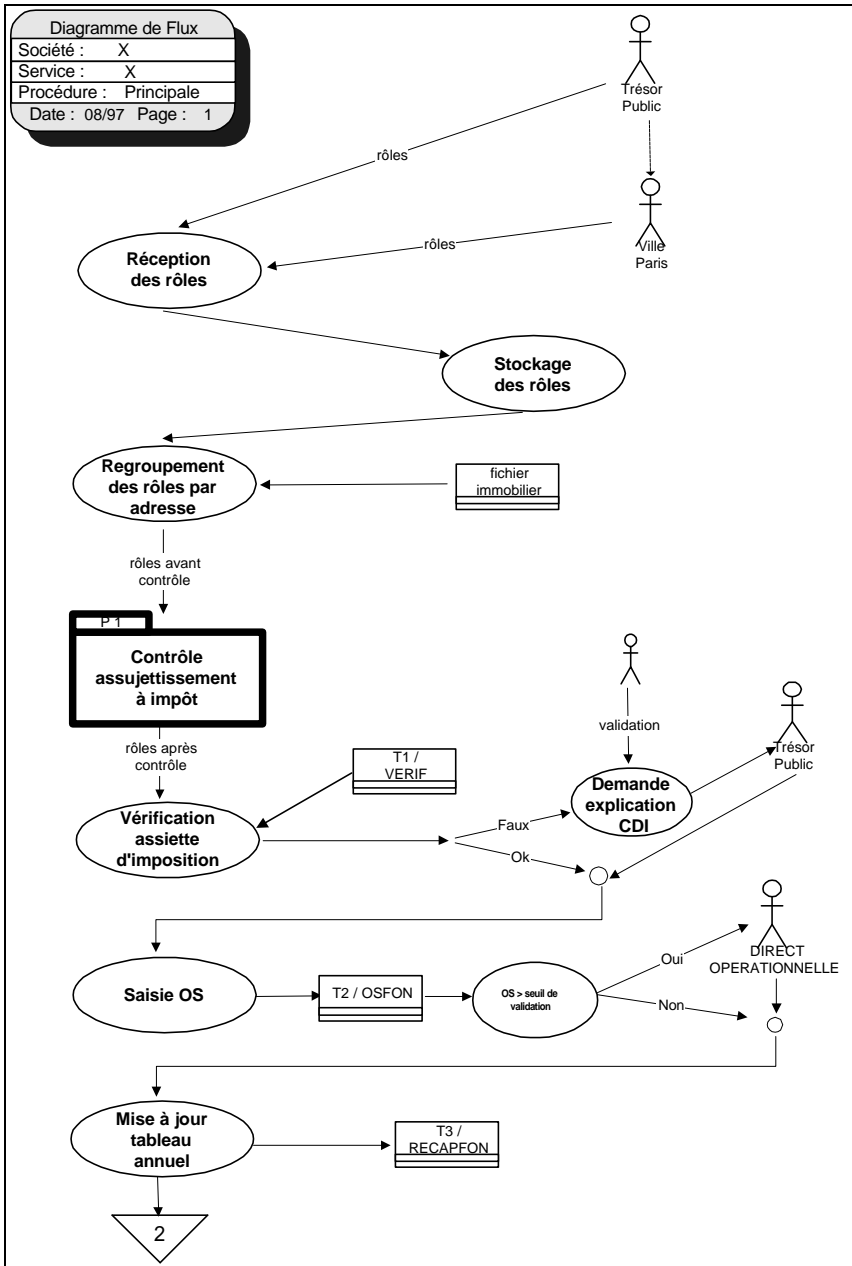


Figure 10. — Notation UML, diagramme de flux de premier niveau

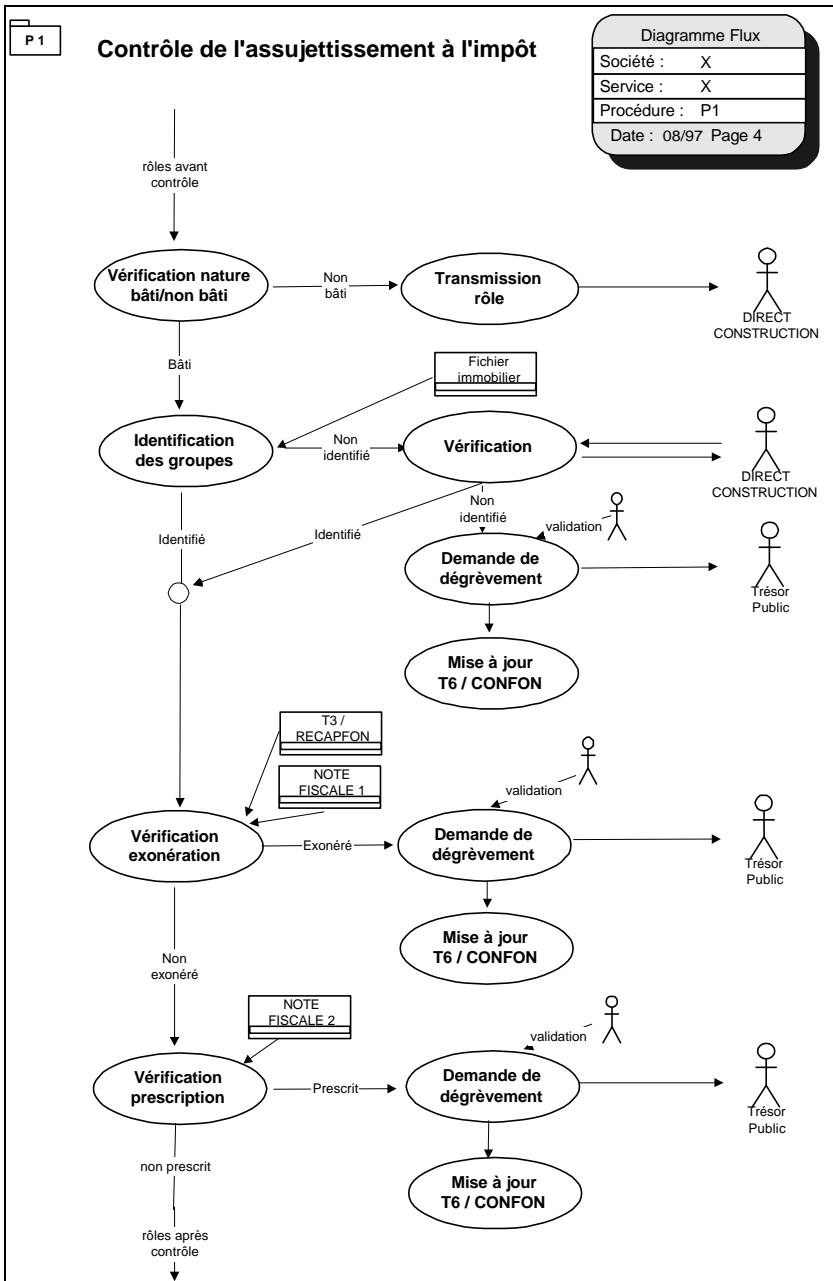


Figure 11. — Notation UML, diagramme de niveau 2 (P1 décomposé)

## 5. Bibliographie des ouvrages

### Bibliographie RAD, conduite de projet

Martin (James), *Rapid Application Development*, Macmillan, 1991.

Vickoff (Jean-Pierre), *RAD - Développement Rapide d'Applications*, MGI 1995

Vickoff (Jean-Pierre), *RAD - Développement Rapide d'Applications*, Macmillan, 1996.

Bouchy (S.), *L'Ingénierie des systèmes d'information évolutifs*, Eyrolles.

Englewood (C.), *JAD the Group Session, Approach to system design*, Prentice Hall, 1991.

Habrias (H.), *La Mesure du logiciel*, Technea, 1995.

Henry (A.), Monkam-Daverat (L.), *Rédiger les procédures de l'entreprise*, Les Éditions d'organisation, 1995.

Hugues (J.), Leblanc (B.), Morley (C.), *RAD - Une méthode pour développer plus vite*, InterEditions, 1996.

Kadima (H.), *Pratique de la méthode SA/SD*, Masson, 1996.

Lorino (P.), *Méthodes et pratiques de la performance - Le guide du pilotage*, Les Éditions d'organisation, 1997.

Mc Carty (J.), *54 Règles d'or pour un grand logiciel*, Microsoft Press, 1997.

McConnell (S.), *Stratégie de développement rapide*, Microsoft Press, 1996.

Moréjon (J.) & Rames (J-R), *Conduite de projets informatiques - Principes et techniques s'appuyant sur la méthodes Merise*, InterEditions, 1993.

Panet (G.) & Letouche (R.), *Merise 2, modèles et techniques Merise avancés*, Les Éditions d'organisation, 1994.

Printz (J.), *Le Génie logiciel*, Presses universitaires de France, 1995.

Silvestre (P.) & Verlhac (D.), *Le Développement des systèmes d'information de Merise à RAD*, Hermès, 1996.

Soberman (M.), *Le Développement rapide d'applications*, Hermès, 1996.

Townsend (R.), *Faites décoller vos hommes et votre entreprise*, Editions du Seuil, 1985.

Wood (J.) & SILVER (D.), *Joint Application Design*, Edition John Wiley, 1989.

*Guide de création d'applications client-serveur avec Visual Basic*, Microsoft Corporation, 1996.

*Ingénierie des systèmes avec Merise, vers une deuxième génération*, Sybex, 1993.

*Modèle d'évolution des capacités logiciel*, Software Engineering Institute, 1993.

*Pratiques clés du Modèle d'évolution des capacités logiciel*, Software Engineering Institute, 1993.

## **Bibliographie Objet**

Budd (T.), *La Programmation par objets*, Edition Addison-Wesley, 1991.

Chappell (D.), *Au cœur de Active X et OLE*, Microsoft Press, 1996.

Jacobson (I.), *Le Génie logiciel orienté objet*, Addison-Wesley, 1993.

Lai (M.), *UML - La Notation unifiée de modélisation objet*, InterEdition, 1997.

Lemay (L.) & Perkins (C. L.), *Le Programmeur JAVA*, Macmillan, 1996

Muller (P. A.), *Modélisation Objet avec UML*, Eyrolles, 1997.

Rumbauch (J.), *OMT Modélisation et conception orientées objet*, Prentice Hall (Masson), 1996.

Silverio (N.), *Programmer en COBOL objet*, Eyrolles, 1996.

Vauquier (D.), *Développement orienté objet, principes, processus, procédés*, Eyrolles, 1993.

Warren (H.), *Active X*, Macmillan, 1996.

## Bibliographie Management - Qualité

- Adam (B.), *L'Analyse de la valeur stimulant des ressources humaines*, ESF, 1987.
- Archier (G) & Sérieyx (H.), *L'Entreprise du 3<sup>ème</sup> type*, Editions du Seuil, 1984.
- Archier (G) & Sérieyx (H.), *Pilotes du 3<sup>ème</sup> type*, Editions du Seuil, 1986.
- Ballay (J-F.), *Capitaliser et transmettre les savoir-faire de l'entreprise*, Eyrolles, 1997.
- Bartoli (A.), *Communication et organisation, pour une politique générale cohérente*, Les Éditions d'organisation, 1994.
- Beaudoin (P.), *La Gestion du changement, une approche stratégique pour l'entreprise en mutation*, Stratégies d'entreprise, 1990.
- Delafollie (G.), *Analyse de la valeur*, Hachette, 1991.
- Enrègle (Y.), *Le Management revisité. Diriger en économie tendue*, Les Éditions d'organisation, 1997.
- Kamiske (G.) & Brauer (J-P.) *Management de la qualité de A à Z*, Masson, 1995.
- Laudoyer (G.), *La Certification, un moteur pour la qualité*, Les Éditions d'organisation, 1993.
- Lamprecht (J. L), *ISO 9000 Se préparer à la certification*, AFNOR, 1994.
- Landier (H.), *Vers l'entreprise intelligente*, Calmann-Lévy, 1991.
- Lefebvre (C.), *Concevoir et conduire un projet de changement*, Les Presses du management, 1997.
- Le Mouël (J.), *Critique de l'efficacité*, Editions du Seuil, 1991.
- Mucchielli (R.), *L'Interview de groupe*, ESF, 1987.
- Petit-Etienne (M.) - Peyraud (Y.), *Reengineering, mode d'emploi*, Les Éditions d'organisation, 1996.
- Sary (P.), *La Stratégie de la programmation neurolinguistique dans l'entreprise*, Editions Retz, 1990.

## Index

### A

abréviation 25  
abstraction 42; 49  
abstraction 52  
acquisition 50  
AGL  
    conception 22  
Amérique du Nord 16; 32; 45  
anecdote 40  
architecture 9; 18; 26; 50  
ATC 31  
autonomie 25

### B

binôme 12  
budget  
    cadrage 21

### C

Canada 40; 45; 47  
caricature 41  
certification 69  
charte graphique 24  
**check-list** 24  
cohérence systémique 13; 21  
communication  
    espace 12; 22; 42  
    modèle 21; 25  
    qualité 10; 18; 19; 32; 43  
    qualité 51  
    réseau 50  
    réseau 55  
    technique 12  
compétitivité 10  
compromis 37; 42  
concepteurs-développeurs 25

consensus  
    RAD 23  
consultant 8; 45  
coordinateur 27  
courbe du soleil 53  
culture  
    changement 10; 45  
    entraide 29; 31  
    France 44  
    RAD 18  
Cybercorp 47

### D

Darwin 46  
De La Palisse 38  
décomposition  
    objet 14  
dérive  
    causes 42  
    méthode 15  
    pratiques 62  
    stat. 8  
*Design to Cost* 16  
deuxième génération 68  
dimension temporelle  
    durée 22  
    phase 11; 42  
    time-box 16; 19  
dynamique  
    applicative 12  
    de projet 13; 19  
    équipe 28

### E

échec  
    causes 39; 42; 46  
    causes 51; 52  
    crise 46; 62

- de projet 10; 11; 15; 36
- responsabilité 37
- statistiques 8; 61
- économie 8; 31; 40
- entité-relation 22
- entraide 13; 29; 31
- entretien
  - de groupe 19; 20; 24
  - discours utilisateur 24
  - modélisation directe 29
- ergonomie 19
- état de livraison permanente 63
- évaluation
  - de projet 25

**F**

- Focus
  - communication 32
  - pratiques 55
  - validation 64
- fonctionnalités réduites 12
- fonctionnalités
  - du projet 30
- fonctionnel
  - cheminement 22; 64
  - conformité 15
  - coordinateur 12
  - couverture 21; 30
  - couverture 53
  - domaine 21; 43
- formation
  - permanente 39
  - programme 19
  - simplifiée 49; 64
  - swat 29
  - swat 52
  - utilisateur 22; 24
- forums 48

**G**

- gaspillage 14; 40; 51
- Groupware 9

**H**

- héritage 16
- hiérarchie de fonctions
  - modélisation 24
  - modélisation 22
  - pratiques 25

**I**

- immersion 21
- ingénierie 61
- ingénierie
  - CMM 62
- intérimaire 40
- ISO 35

**J**

- jalons ZD 32
- James Martin 16; 18; 27; 34; 47; 67

**L**

- lancement du projet 21
- livrable 19; 63
- livrable 52; 54

**M**

- maintenance
  - applicative 31
  - module 29
  - prévue 29
- Maîtrise
  - d'Ouvrage 24; 25
- MERISE
  - et RAD 15; 16; 42
  - méthode 13
  - SDMS 53
- méthode de conduite de l'évolution 13
- methodologique 24
- mutation
  - méthodes 39
  - métiers 13; 26
  - outils 39

**N**

neutralité 43  
noyau 45

**O**

OCX 15; 30; 50  
outil  
  achat 50  
  construction 22  
  évaluation 16  
  présentation 24

**P**

participation  
  équilibre 18  
  M.O. 49  
pérennité 50  
périmètre  
  général 19  
  modélisation 28  
pilote  
  site 22  
  site 55  
planification  
  cadrage 63  
  outils 24  
  pratiques 30; 33; 62  
poids des phases 23  
présentation  
  prototype 22  
processus  
  adaptation 9  
  qualité 27; 33; 47  
  RAD 24; 34  
  RAD/CMM 62  
  réingénierie 21  
productivité  
  clés 30  
  évaluation 46; 63  
  problèmes 23; 45  
  RAD 51  
projet type 38  
prototypage

actif 29  
actif 55  
construction 20  
modélisation 23  
RAD 18  
spécifications 13  
prototype  
  niveau 19; 22  
  validation 20  
publication 32

**R**

raccourci  
  abstraction 42  
  méthode 52  
  tentative 62  
rapporteur 24  
réactivité 9  
réactivité  
  intervenants 49  
  optimisation 63  
  organisation 47  
recette  
  finale 64  
  initiale 22  
  partielle 20  
  pratiques 42  
récompense 29; 31  
reengineering 25  
réingénierie 21  
relations interpersonnelles 12; 28  
ressource  
  disponibilité 30  
  gaspillage 14; 51  
  priorisation 21  
  profil 28; 45; 46  
rétroprojecteur 24  
réutilisation 15

**S**

sacerdoce 43  
savoir-faire 61  
secrétaire 40  
secrétaire 55

SEI 62	
sessions parallèles 22	
sous-ensemble vidéo 25	
spécialiste-généraliste 13	
spécification	
détaillée 13	
SPICE 15; 35	
stratification 9	
structuration	
données 49	
phasage 11; 16	
phase 19	
structure	
(Top-Down) 13	
surqualification 46	
SWAT	
composition 13	
culture 29; 31; 32	
organisation 19	
synergie 12; 63	
systémique	
cohérence 21	
conception 13; 14; 49	
organisation 9	
	<b>T</b>
	temps partiel 30
	thèmes
	découpage 19
	use-case 64
	<i>Time Boxing</i> 16
	<i>turnover</i> 23
	typologie
	projet 35; 38
	raccourci 52
	variée 42
	<b>U</b>
	use case 64
	utilisateur
	engagement 12; 23; 25; 40
	exigences 19
	use case 64
	<b>V</b>
	verrouillage 21; 25
	Visual Basic 40

## Table des illustrations

<i>Figure 1. — Jalons décisifs du cycle projet RAD</i>	20
<i>Figure 2. — Parallélisation et sérialisation des grands projets</i>	21
<i>Figure 3. — Eléments de la problématique "taille du SWAT"</i>	31
<i>Figure 4. — Facteurs d'échec recensés - 1</i>	36
<i>Figure 5. — Facteurs d'échec recensés - 2 -</i>	37
<i>Figure 6. — Comparatif des cycles de développement</i>	42
<i>Figure 7. — Dilemme classique de l'animateur RAD</i>	44
<i>Figure 8. — Dimension organisationnelle du flux qualité</i>	48
<i>Figure 9. — CMM : 5 paliers dans la rigueur</i>	62
<i>Figure 10. — Notation UML, diagramme de flux de premier niveau</i>	65
<i>Figure 11. — Notation UML, diagramme de niveau 2 (P1 décomposé)</i>	66

# Transfert d'expertise

## Conférence - formation

Jean-Pierre Vickoff vous propose une journée basée sur la complémentarité de trois modules successifs :

- Le premier d'une durée de 15 à 30 minutes a pour but de sensibiliser les **décisionnaires** aux avantages du reengineering.
- Le second d'une durée de 90 mn présente les apports et contraintes du RAD à la **maîtrise d'ouvrage**, aux **correspondants informatiques** et aux **utilisateurs (le principe des sessions RAD, les techniques d'entretiens, la qualité et la durée de l'engagement)**.
- Le troisième (après-midi) est un perfectionnement dédié aux **informaticiens de la maîtrise d'oeuvre**. Il couvre **outre les aspects organisationnels et humains tous les axes techniques de la conduite de projet en conception et en développement**.

## Conseil, support et services

- Tuning de projets (évaluation, recentrage et solutions alternatives)
- Conduite de projets sous contraintes (qualité, délais, ressources)
- Optimisation :
  - **cadre méthodologique**
  - **méthode de communication**
  - **techniques de conception**
  - **techniques de développement**
- Formation et accompagnement du changement des conditions de développement
- Support en intégration technologique et nouvelles architectures (client-serveur, internet / intranet, 3-tiers)
- Mise en œuvre des solutions Microsoft et Oracle (outils "Visual Studio", InterDev, etc).