Nouveaux regards sur les méthodologies d'analyse, de conception et de programmation informatiques

Frédéric ADAM et Brian FITZGERALD
Enseignants en informatique de gestion - University College Cork, Ireland

RÉSUMÉ

Un examen critique des théories dominantes dans le domaine de la conception et de la programmation des systèmes informatiques révèle que la plupart des outils, des techniques et des méthodologies qui sont utilisés dans ce domaine sont fondés sur des concepts qui datent de la période 1967 / 1977. Les cycles de vie de type "cascade" (System Development Life Cycle ou SDLC), le prototypage, la décomposition algorithmique, le masquage d'information, et les techniques d'analyse structurée telles que les diagrammes de flux de données (Data Flow Diagrams ou DFD) et les diagrammes d'entité-relation (Entity Relationship Diagrams ou ERD) datent tous de cette période. L'approche orientée objets elle-même trouve ses origines dans cette période.

Cet article présente un certain nombre d'arguments qui militent en faveur d'une révolution dans le domaine de la réalisation de projets informatiques et de la création de nouvelles méthodologies mieux adaptées aux besoins actuels. Nous basant sur une série d'interviews d'informaticiens expérimentés et sur une enquête postale, nous présentons des observations qui révèlent le profil très différent des projets informatiques actuels par rapport à ceux qui furent caractéristiques de la période 67 / 77. Finalement, nous tentons d'expliquer pourquoi tant la recherche que la pratique n'ont pas avancé aussi vite qu'elles auraient dû dans le domaine de l'informatique.

Mots-clés : Méthodes de développement de systèmes, Rapid Application Design, Profil des développements informatiques actuels.

ABSTRACT

A critical examination of currently accepted theories in the systems analysis and design area reveals that the great majority of tools and techniques used for the development of information systems have emerged in the 1967 / 1977 decade. The numerous Systems Development Life Cycles (SDLCs) in existence, prototyping, algorithmic decomposition, data hiding and structured analysis techniques such as Data Flow Diagrams and Entity Relationship diagrams all originated in this period. Even the Object Oriented approach can be traced back to this "golden decade".

This article presents a number of arguments in favour of a revolution in the development of information systems and pointing to the need for the creation of new development methodologies more suited to the current needs of organisations. Based on a series of interviews with experienced systems developers and a postal survey of Irish organisations, we show how the profile of today's IS projects is now radically different from the projects of the 67 / 77 period. Finally, we endeavour to explain why both practice and research have not progressed as fast as they should have in the IS area.

Key-words : IS development methods, Rapid application design, Profile of today's IS projects.

1. INTRODUCTION

Se basant sur un certain nombre d'observations qui montrent que l'industrie du logiciel subit une crise (caracterisée pas la chute des prix du matériel informatique et l'impossiblité de répondre à la demande en applications), Cox (1990) a envisagé la nécessité d'une "révolution industrielle" dans la conception et la réalisation des systèmes informatiques et la création de nouvelles approches mieux adaptées aux besoins de notre environnement en mutation permanente. Deux arguments permettent de mieux cibler ce débat. Premièrement, il est remarquable que la plupart des méthodologies et des techniques de développement informatiques disponibles à l'heure actuelle sont basées sur des concepts maintenant anciens. Deuxièmement, il est évident que le profil des projets informatiques avec lesquels les informaticiens sont aux prises a évolué de façon particulièrement radicale au cours des 10 dernières années et que l'environnement industriel et commercial dans lequel ces projets se déroulent a lui aussi beaucoup changé.

Cet article retrace les origines des concepts sur lesquels sont fondées les méthodologies actuelles et les replace dans leur contexte historique. Parce que l'approche orientée objets a été identifiée par plusieurs auteurs comme le prochain paradigme pour le développement des systèmes informatiques (une analyse que nous ne partageons pas forcément), une attention particulière lui est accordée. Nous sommes ainsi amenés à nous demander dans quelle mesure l'approche orientée objets est une révolution ou une simple évolution des méthodes existantes. Dans la section suivante, l'article présente une série d'observations qui illustrent le profil des projets informatiques d'aujourd'hui sur la base d'une enquête postale et de 21 interviews plus qualitatives avec des informaticiens expérimentés conduites par l'un des auteurs dans dix entreprises différentes. Finalement, l'article tente d'expliquer pourquoi, tant la théorie que la pratique sont décalées par rapport à cette réalité du développement informatique et offre des suggestions pour la création de méthodologies futures.

2. REGARDS HISTORIQUES SUR L'ANALYSE ET LA CONCEPTION DES SYSTÈMES INFORMATIQUES

La plupart des méthodologies de conception et de programmation (Systems Development Methods ou SDM) utilisées à l'heure actuelle dans la réalisation des projets informatiques ont leur origine dans des concepts qui ont émergé au cours de la décennie 67 / 77. Par exemple, les cycles de vie des projets de développement (SDLC) qui trouvent leur origine dans la théorie générale des systèmes des années 30 (Enger, 1981) ont été introduits à cette période dans la gestion des projets informatiques (Royce, 1970). Ces SDLC furent à leur tour à la base de la plupart des méthodes (Davis *et al.*, 1988 ; Orr, 1989) et ont été décrits comme étant la caractéristique essentielle de tous les projets de développement informatique (Ahituv *et al.*, 1984). De même, le prototypage fut utilisé par un nombre grandissant de développeurs de

systèmes pendant cette période pour éviter les problèmes inhérents aux traditionnels SDLC en "cascade" (Agresti, 1986). Le prototypage fait maintenant partie intégrante de nombreuses méthodologies commerciales de développement (Boehm, 1988, Downs *et al.*, 1992).

En ce qui concerne l'approche structurée, qui a été décrite comme étant la plus fréquemment utilisée en Amérique du Nord et en Europe (Yourdon, 1991), le concept de masquage d'information, qui stipule que chaque module doit représenter un processus et un seul et doit révéler aussi peu que possible les données qu'il utilise (Parnas, 1972), le concept de décomposition algorithmique, c'est-à-dire la division progressive des fonctions primaires d'un logiciel en sous-fonctions jusqu'au niveau le plus élémentaire, et les concepts de cohésion et de couplage ont tous émergé pendant cette période (De Marco, 1978 ; Stevens, Myers et Constantine, 1974). La méthode Merise qui est la plus utilisée en France est apparue juste à la fin de la période considérée puisqu'elle a été lancée en 1977 par le ministère de l'Industrie (Avison et Fitzgerald, 1995). De plus, la plupart des techniques de représentation qui se retrouvent dans toutes les méthodologies, les diagrammes d'entité-relation (Chen, 1976), diagrammes de flux de données et les dictionnaires de données (De Marco, 1978) datent aussi de la période 67 / 77.

SSADM (Structured Systems Analysis and Design Method) qui est la méthodologie la plus utilisée en Grande Bretagne et en Irlande a ses antécédents dans la méthode MODUS utilisée entre 1965 et 1977, tandis que l'approche de Jackson (JSP ou Jackson System Programming) est fondée sur le principe de Bohm et Jacopini (1966).

Dans des travaux plus récents, certains auteurs ont affirmé que l'approche orientée objets représente un nouveau paradigme pour le développement des systèmes (Thomann, 1994), et, en effet, l'approche OO a pris une grosse importance en informatique. Il n'en reste pas moins que le concept d'OO n'est pas aussi révolutionnaire qu'il y paraît. Il semble que dans beaucoup de cas, les méthodologies dites OO ne sont que des évolutions de méthodologies existantes ayant elles aussi leur origine dans les années 60. Pour être tout à fait précis, l'approche orientée objets a ses origines dans le langage de programmation Simula qui fut utilisé en Norvège à partir de 1967. Avant cela, certains principes orientés objets tels que l'encapsulation ont été utilisés dans la conception du missile Minuteman dès la fin des années cinquante (Dyke et Kunz, 1989).

Ainsi, bien que l'intérêt de la profession informatique pour l'approche orientée objets soit bien réel et que les prédictions les plus optimistes aient été faites concernant la rapidité d'adoption de ces méthodes par les informaticiens (Topper, 1992), on peut s'interroger sur le caractère révolutionnaire de l'approche orientée objets. L'encapsulation, qui permet d'enrober des données et des programmes dans un objet est très semblable au concept de masquage d'information (Parnas, 1972). De même, la communication entre objets s'effectue sous forme de messages qui passent d'un objet à un autre pour déclencher l'exécution d'une procé-

dure. Ce mécanisme rappelle le principe de couplage. En fait, sur la trentaine de méthodologies orientées objets qui ont été documentées (Firesmith, 1993), beaucoup sont simplement des évolutions d'anciennes méthodes bien connues.

Néanmoins, certains auteurs ont affirmé que l'approche OO est probablement l'approche la plus appropriée pour le développement des systèmes informatiques futurs étant donné les besoins des nouvelles formes organisationnelles (Bessagnet *et al.*, 1994). Il semble qu'il y a des avantages certains à appliquer l'approche OO dans ces nouveaux contextes. En particulier, il a été suggéré que l'approche OO s'applique naturellement aux éléments du monde réel grâce à son pouvoir de modélisation et aux possibilités de réutilisation des programmes qu'elle permet.

La réalité de l'utilisation des approches OO est cependant loin de correspondre au volume des publications consacrées à cette approche. La plupart des recherches et développements dans le domaine OO ont été effectués en dehors du cadre des développements informatiques commerciaux dont les entreprises ont besoin et l'application de l'approche OO à de vrais projets informatiques commerciaux semble bien problématique. L'étude de Firesmith (1992) qui portait sur 23 projets utilisant un langage de programmation OO a conclu qu'aucun projet n'avait pu bénéficier totalement des avantages supposés d'OO. En guise de conclusion, Firesmith (1992) propose un nombre de conseils pratiques pour aider les entreprises à appliquer les principes d'OO, mais il précise aussi que les bénéfices ne seront visibles qu'après plusieurs projets et qu'en fin de compte, les problèmes de gestion des projets sont plus importants que les problèmes techniques.

Ainsi, en dépit du potentiel des approches OO, on peut se demander si les avantages procurés par l'approche OO seront un jour à la hauteur des prédictions faites dans les revues scientifiques.

3. LE DÉVELOPPEMENT INFORMATIQUE DANS LES ANNÉES 90

Les sections précédentes de cet article ont indiqué que la plupart des méthodes de développement utilisées à l'heure actuelle, y compris les méthodes OO qui sont pourtant présentées comme nouvelles, sont fondées sur des concepts étant apparus entre 1967 et 1977. Pour achever de montrer la nécessité de changements radicaux dans le domaine du développement des systèmes informatiques, il nous reste à décrire les profondes différences qui existent entre l'environnement dans lequel les systèmes informatiques d'aujourd'hui sont développés et l'environnement des années 67 à 77. Nous tentons de montrer ces différences en nous fondant sur des publications récentes, sur une étude statistique et sur une série d'interviews conduites par l'un des auteurs (Fitzgerald 1994, 1996, 1997). Les questions qui sont soulevées dans les sections suivantes sont liées à la nature évolutive du monde des entreprises, la nature changeante des contextes organisationnels dans lesquels les systèmes informatiques sont développés et à l'existence de fortes pressions pour des dévelop-

pements de plus en plus rapides (comme en témoigne l'apparition du concept de Rapid Application Development ou RAD).

3.1. L'Univers commercial du développement informatique

Les entreprises d'aujourd'hui doivent faire face à un environnement qui évolue de plus en plus vite ; phénomène qui a été constaté par de nombreux auteurs. Rockart et De Long (1988) ont fait référence au "métabolisme accéléré du commerce d'aujourd'hui" qui requiert que les entreprises réagissent plus efficacement et plus rapidement. Baskerville *et al.* (1992) ont aussi insisté sur les conséquences des modifications du monde actuel sur le développement des systèmes. Ils indiquent que les méthodes de développement structurées sont tournées vers des projets de développement de grande taille et de longue durée, ce qui n'aide pas les entreprises à résoudre efficacement les problèmes posés par l'évolution rapide des marchés et des produits. Cela pousse les entreprises à se tourner vers des solutions de plus en plus de court terme. Dans cette situation, il n'est plus ni rentable ni même possible de suivre les étapes formelles des méthodes de développement de systèmes 'en cascade'. Baskerville *et al.* (1992) sont allés plus loin et ont affirmé que l'utilisation des méthodes traditionnelles est devenue un frein à la résolution rapide des problèmes informatiques des entreprises. Ainsi, les développeurs ne peuvent plus se permettre d'appliquer patiemment les vieilles méthodes et en sont réduits à choisir entre ce qu'ils ont le temps de faire et ce qui doit être abandonné (Brown, 1985).

3.2. Stratégies de remplacement pour le développement de systèmes

Cette évolution de l'environnement commercial pousse les entreprises à rechercher des stratégies de remplacement pour faire face à leurs besoins en systèmes informatiques. Bansler et Havn (1994) ont identifié une nouvelle tendance dans l'industrie du logiciel. Il semble que les entreprises comptent de moins en moins sur leurs services informatiques internes pour le développement de systèmes et achètent en contrepartie des logiciels clef en main ou utilisent l'outsourcing comme stratégie de remplacement. Ils ont appelé cette tendance "l'industrialisation du développement informatique" et ont suggéré que la même mutation a déjà eu lieu dans d'autres branches plus anciennes de l'ingénierie. Bansler et Havn (1994) ont aussi décrit les conséquences pratiques de cette mutation de l'industrie du logiciel. Par exemple, de nombreux utilisateurs assemblent leurs systèmes sur la base de modules disponibles sur le marché et les intègrent avec des morceaux de logiciel développés spécialement pour respecter les particularités et habitudes de l'entreprise. Bien entendu, les SDM traditionnelles ne sont d'aucun secours pour la gestion et l'organisation de ces nouveaux processus de création de systèmes et leur application à de tels exercices informatiques en est ainsi d'autant plus difficile et dangereuse. Dans l'état de nos connaissances, il semble que seule l'étude de Nilsson (1990) se soit penchée sur les problèmes

méthodologiques posés par ce type de projets informatiques. En l'absence de méthode(s) rigoureuse(s) pour la conduite de tels projets, les entreprises désireuses d'acquérir des systèmes existants doivent inventer des processus de selection et d'adaptation pour augmenter leurs chances de succès ; exercice difficile et risqué comme l'ont prouvé certains exemples notoires (cf. SNCF et son système de réservation SOCRATE ; décrit dans Adam et Cahen, 1997).

3.3. Etude empirique du nouveau profil des projets informatiques

Pour confirmer les tendances qui sont décrites dans les sections 3.1 et 3.2 et en rendre compte de façon plus précise, une approche plus empirique est cependant nécessaire. Dans cette section, nous présentons donc les résultats d'une enquête récente de l'industrie du logiciel qui illustrent le profil des développements informatiques actuels et montrent à quel point le profil du projet typique s'est modifié. Un questionnaire fut envoyé à 776 informaticiens (du programmeur au directeur informatique) dans autant d'entreprises irlandaises. Sur le total des 186 réponses reçues, 162 purent être utilisées dans l'analyse qui suit, ce qui donne un taux de réponse global de 24 %. Un tel taux de réponse est similaire à ceux obtenus dans d'autres enquêtes postales sur l'utilisation des méthodes de développement de systèmes au Royaume-Uni (Hardy, Thompson et Edwards ont obtenu 20 % et 29 % dans leurs études en 1995) et aux Etats-Unis (Mahmood a obtenu 18 % en 1987 et Palvia et Nosek ont obtenu 22 % en 1993).

Le tableau 1 donne une indication des caractéristiques des organisations qui forment notre échantillon. Il s'avère que la plupart des organisations étudiées comptaient entre 100 et 1 000 employés (43 %) et avaient un service informatique de petite taille - soit de 1 à 5 persones (44 %). Ces statistiques nous permettent de conclure qu'une grande proportion de ces organisations de taille moyenne étaient pourvues de services informatiques restreints. En effet, ces chiffres sont nettement en dessous de ceux rapportés par des études précédentes comme celle de Sumner et Sitek (1986). Ils sont en revanche en accord avec l'étude de Friedman (1989) qui a identifié une diminution de la taille des départements informatiques au cours des années 80. Friedman en avait conclu que cette évolution était due à une décentralisation des informaticiens dans les autres services et au recours grandissant à l'outsourcing. Il est à noter que cinquante des entreprises étudiées (31 %) étaient des sociétés de conseil en informatique ou des SSII.

Le tableau 2 présente les résultats de l'étude qui concernent le niveau d'expertise des répondants et le profil des projets informatiques sur lesquels ils travaillent couramment. Il est à noter que le niveau d'expérience des développeurs ayant accepté de participer à l'étude (plus de 15 ans en moyenne) est assez élevé pour conférer une validité suffisante aux résultats obtenus.

Le tableau 2 montre, entre autres, le haut niveau d'utilisation des logiciels clef en main (environ 40 %) et le niveau significatif d'outsourcing (environ 13 %). Sur la base de ces chiffres, on peut

Industrie	%	(n)	Nombre total d'employés	%	(n)	Nombre d'employés Service info.	%	(n)
Consultant / SSII	31 %	(50)	1 à 10	12 %	(19)	1 à 5	44 %	(71)
Secteur public / Education	7 %	(11)	10 à 100	29 %	(47)	5 à 20	30 %	(49)
Ent. industrielle / Distribution	36 %	(58)	100 à 1 000	43 %	(70)	20 à 100	19 %	(31)
Grossiste / Détaillant	4 %	(7)	1 000 à 5 000	9 %	(15)	> 100	7 %	(10)
Finance / Assurance	12 %	(19)	> 5 000	7 %	(11)			
Service / Communication	6 %	(10)						
Autres	4 %	(7)						

Tableau 1 : Profil démographique des organisations étudiées

	Pourcentage ou Moyenne	
Experience des répondants (en nombre d'années)	15,07	
Profil des développements :		
% développements internes	47	
% développement "outsourcé"	13	
% utilisation de logiciels "clef en main"	40	
Profil des projets informatiques :		
Nombre de développeurs	3,5	
Durée (en mois)	5,7	
Hardware utilisé	%	(n)
Mainframe	20 %	(28)
Mini	26 %	(36)
PC	33 %	(41)
Combinaison / Autres	21 %	(30)

Tableau 2 : Profil des développements informatiques actuels

conclure que le développement interne n'est plus aussi important que par le passé pour un grand nombre d'entreprises. De même, le fait qu'un projet informatique typique rassemble une équipe de trois ou quatre développeurs et dure moins de six mois montre le profil très différent des projets actuels - plus courts et plus rapides - qui contraste avec les résultats d'études précédentes comme celle de Jenkins *et al.* (1984) par exemple. Les différences que nous avons relevées avec l'étude de Jenkins sont importantes parce que cette étude date du début des années 80

(c'est l'une des plus anciennes de ce type) et parce qu'elle est reconnue comme l'une des plus couramment citée par les auteurs dans ce domaine.

On peut s'interroger sur les raisons de cette rapide évolution et se demander si cette fragmentation des projets ne reflète pas une adhérence plus stricte aux principes de base des méthodologies traditionnelles, en l'occurrence "diviser pour mieux régner" ("divide and conquer"). Cela semble en tout cas indiquer une approche plus réaliste de la part des chefs de projet qui sont conscients que les gros investissements et la prise de risques inconsidérés ne sont plus de mise. Un autre résultat notable de cette étude concerne la plate-forme la plus utilisée pour les développements informatiques : la plupart des projets étudiés étaient des applications destinées à opérer sur des PC ; plate-forme réservée jusqu'ici à de petits logiciels génériques (feuille de calcul, traitement de texte, etc.).

L'étude indique également que 60 % des participants n'utilisent pas de méthode de développement identifiable. La raison la plus fréquente de cette non-utilisation des méthodes existantes vient de la perception des développeurs de l'inadéquation de ces méthodes au type de développement informatique dont leurs entreprises ont besoin.

Le tableau 3 indique le détail des résultats de notre étude qui concernent le degré d'utilisation réel des méthodologies traditionnelles dans les entreprises étudiées. La colonne de gauche confirme que 60 % des répondants n'utilisent aucune SDM. Les 40 % restants appliquent une SDM à la lettre (14 %) ou sous une forme modifiée en fonction de leurs besoins (12 %) ou encore utilisent une méthode tout à fait personnelle (14 %).

Total n'utilisant aucune SDM	**Total utilisant une SDM**	**Parmi ceux qui utilisent une SDM**		
		Utilisent une SDM commerciale[1]	**Utilisent une SDM basée sur une SDM commerciale**	Utilisent une SDM personnelle
60 % (97)	40 % (65)	14 % (23)	12 % (20)	14 % (22)
		SSADM (8) IE (6) Oracle (4) Method 1 (3) OMT (2) LSDM (1) STEP (1) STRADIS (1) Progen (1)	SSADM (9) IE (3) etc.	

[1] *(n = 23, mais certains répondants ont donné plus d'une réponse).*

Tableau 3 : Degré d'utilisation réel des méthodologies traditionnelles

Ces résultats indiquent que relativement peu d'entreprises irlandaises utilisent les méthodologies commerciales habituelles. Les chiffres que nous avons obtenus (14 % des entreprises appliquent à la lettre une méthode connue) sont en effet très inférieurs à ceux rapportés dans des études précédentes. Par exemple, l'étude de Jenkins *et al.* (1984) qui sert souvent de réference en la matière mentionne des taux d'utilisation de SDM de l'ordre de 85 %, mais une étude irlandaise qui date de 1992 avait déjà mesuré un taux de 38 % (O'Duffy, 1992). On peut donc se demander s'il existe une différence fondamentale entre les taux d'utilisation des SDM aux Etats-Unis (étude de Jenkins), en Europe et plus particulièrement en Irlande. Notre étude ne permet pas de conclure sur ce point et il serait intéressant d'étendre l'étude des taux d'utilisation à d'autres pays d'Europe afin de mesurer les effets d'éventuelles "préférences nationales".

L'une des nouveautés de notre étude consistait donc à s'intéresser plus spécialement aux entreprises qui utilisent des SDM modifiées. Parmi celles-ci nous avons relevé une certaine confusion de la part des répondants qui ont avancé des noms de méthodes qui nous étaient totalement inconnus ou qui ne pouvaient être considérés comme des SDM à proprement parler. Un certain nombre de répondants ont même cité ISO 9000 comme la base de leur méthode interne, ce qui laisse à penser qu'un petit nombre d'informaticiens ne sont pas très sûrs de ce qu'une méthodologie de développement représente.

Nous avons utilisé les résultats présentés dans le tableau 3 pour contraster le degré d'usage que les entreprises étudiées font des outils et des techniques normalement associés avec les méthodologies traditionnelles. Les répondants devaient ainsi indiquer s'ils utilisaient couramment le prototypage, les diagrammes de flux de données ou d'autres techniques très connues (voir tableau 4). Parmi ces derniers, le prototypage, les diagrammes de flux de données, les dictionnnaires de données et les diagrammes d'entité-relation sont les plus populaires à la fois dans les entreprises utilisant une SDM et dans celles qui n'en utilisent pas. Le résultat le plus remarquable réside dans le fait que les entreprises qui suivent une SDM s'avèrent faire un usage bien plus important de **tous** les outils et techniques proposés. Cela semble confirmer que les SDM aident effectivement les informaticiens à coordonner leur utilisation des outils et techniques traditionnels.

Ces résultats sont à comparer avec ceux du tableau 5 qui indique la proportion de l'effort de développement que les développeurs allouent à chacune des phases et activités qui composent les projets informatiques. Comme dans le tableau précédent, ces résultats sont contrastés en fonction de l'utilisation (ou non) d'une SDM.

Les résultats présentés dans le tableau 5 indiquent clairement que l'apport des SDM n'apparaît pas être aussi marqué que pour l'utilisation des outils et techniques traditionnels (résultats du tableau 4). En fait, c'est plutot le contraire : la différence entre les entreprises utilisant une SDM et celles qui n'en utilisent pas n'est pas significative statistiquement parlant. Même s'il semble que,

	Entreprises utilisant une SDM	Entreprises n'utilisant pas de SDM
Outils/Techniques		
Joint Application Design (JAD)	31 %	20 %
Prototypage	75 %	57 %
Diagramme de flux de données	71 %	37 %
Diagramme d'entité-relation	63 %	19 %
Diagramme de changement d'état	19 %	6 %
Ordinogramme	55 %	35 %
Dictionnaire de données	74 %	34 %
Process Mini-Specifications	40 %	25 %
Structured Walkthrough	48 %	23 %
Autre	9 %	5 %

Tableau 4 : Usage des outils et techniques traditionnels en fonction de l'utilisation (ou non) d'une SDM

	Temps de développement moyen alloué dans les entreprises utilisant une SDM	Temps de développement moyen alloué dans les entreprises n'utilisant pas de SDM
Activité		
Planification	10 %	10 %
Analyse	17 %	14 %
Design	15 %	12 %
Programmation	28 %	31 %
Test	17 %	17 %
Installation	8 %	10 %
Evaluation	3 %	4 %
Autre	2 %	1 %

Note : Ces résultats ont été obtenus en faisant la moyenne des pourcentages que les participants à l'étude ont eux-mêmes déclaré avoir alloué à chaque activité.

Tableau 5 : Allocation de l'effort de développement entre les différentes phases dans les entreprises utilisant (ou non) une SDM

dans les entreprises qui utilisent une SDM, plus de temps est alloué aux phases d'analyse et de design, les variations sont minimes. Cela nous conduit à nous demander si les méthodologies traditionnelles aident vraiment les développeurs à se concentrer sur les phases importantes comme cela a été suggéré (Ahituv *et al.*, 1984, Baskerville *et al.*, 1992).

Certains auteurs ont affirmé que l'ignorance relative des informaticiens en entreprise était l'obstacle le plus important à une utilisation plus complète des SDM (Ward, 1991). Cette hypothèse n'a cependant pas été vérifiée dans le volet de cette étude réalisée par l'un des auteurs sur la base d'interviews avec des informaticiens expérimentés dans plusieurs entre-

prises (Fitzgerald, 1994). En fait, même les participants qui n'utilisaient pas formellement de SDM avaient une connaissance utilisable de plusieurs méthodes et avaient une certaine expérience de l'utilisation des méthodes commerciales les plus connues. C'était donc par choix qu'ils n'utilisaient aucune méthode, choix motivé par la nécessité de rester flexible et de développer leurs systèmes rapidement. Ainsi, les SDM sont perçues comme potentiellement bénéfiques pour la documentation et l'application des règles et des standards, mais ces avantages ne suffisent pas à contrebalancer les problèmes (et les coûts) liés à l'allongement de la durée des projets.

Un des participants à l'étude a suggéré qu'une approche contingente était plus efficace pour le développement de systèmes informatiques. Selon lui, la majorité du travail de développement s'effectue "dans la tête des analystes programmeurs" où émergent toutes les solutions aux problèmes qui se posent. La tâche la plus importante consiste donc à communiquer ces solutions entre collègues. Ce participant ajouta qu'il existe de telles différences de compétence et de créativité entre les programmeurs qu'il est nécessaire de distribuer les tâches au sein de l'équipe en fonction des possibilités de chacun et d'adapter la communication des spécifications du système aux différents individus (le degré de formalisation des spécifications par exemple).

Ce type d'approche a été appelé "quick-and-dirty" (à la va vite) par certains auteurs (Guimaraes, 1985), mais cela ne correspond pas nécessairement à la réalité de ces méthodes spontanées et non-formulées qui s'avèrent parfois très efficaces comme nous l'avons constaté au cours d'interviews avec des développeurs. Dans une des entreprises étudiée, nous avons trouvé un excellent travail d'équipe fondé sur l'utilisation de technologies de communication telles qu'E-mail et Lotus Notes. Ce processus était très dynamique et différents programmeurs assumaient le rôle de leader à différents stades du projet. Il serait donc intéressant d'étudier comment les développeurs de systèmes pourraient bénéficier de l'application des nouvelles technologies de communication. Cette direction de recherche a déjà été identifiée par Larrasquet et Clare (1996). En conclusion, il faut insister sur le fait que, dans la plupart des cas, les informaticiens ont rejeté les méthodes existantes pour des raisons basées sur leur expérience et non par ignorance. Il s'avère donc que leur expérience et leur connaissance des réalités du développement informatique sont des atouts aussi importants pour le succès des projets informatiques que la bonne utilisation d'une méthode quelconque.

3.4. Comment accélérer le processus de développement des applications ?

On peut donc en conclure que les développeurs de systèmes informatiques en entreprises sont en permanence à la recherche de solutions à même d'augmenter la productivité des activités de développement, qu'il s'agisse de la phase d'analyse, de la phase de design ou de l'implantation. Cette recherche est partagée par les auteurs qui s'intéressent au développement rapide des systèmes (Ra-

pid Application Development ou RAD). Folkes et Stubenvoll (1992) affirment que les besoins des entreprises nécessitent une accélération des processus de développement. D'autres auteurs ont même estimé que la productivité des développeurs de systèmes informatiques devrait être multipliée par 10 pour véritablement servir les besoins des entreprises (Verity, 1987). Hough (1993) s'est joint à ceux qui dénoncent la faiblesse de la philosophie des approches monolithiques de longue haleine :

> Dans le passé, cette philosophie a engendré des processus de développement informatique très longs et rigides qui se sont traduits par du software qui était tout sauf "soft". La plupart des applications sont développées sous la conduite d'approches monolithiques alors que les besoins des entreprises sont évolutifs et changent rapidement(1).

Hough (1993) cite les résultats d'une étude menée par IBM qui a étudié les problèmes qui se posent quand des méthodologies "monolithiques" sont utilisées. Ceux-ci incluent : un ralentissement du processus de développement, pour les programmeurs une tendance à faire durer les projets pendant des années et à en faire une carrière, un certain degré d'ignorance des problèmes organisationnels qui doivent être résolus, une perte d'intérêt de la part des développeurs et une perte de motivation générale des participants au projet. Hough (1993) soutient qu'une méthode axée sur une livraison rapide des systèmes développés pourrait résoudre tous ces problèmes. Son modèle, inspiré d'un modèle de production japonais, a aussi été utilisé par De Grâce et Stahl (1990) dans leur discussion des alternatives qui s'offrent aux développeurs de systèmes (cf. Takeuchi & Nonaka, 1986). L'objectif principal de cette livraison rapide est de produire des résultats à intervalles réguliers, par exemple en livrant une ou plusieurs fonctions tous les mois. Ce concept de livraison régulière est aussi à la base de l'approche de Gilb (1988), appelée "incremental engineering approach". D'après Hough, les avantages d'une livraison rapide des systèmes informatiques sont nombreux et importants dans le contexte d'une entreprise commerciale. Ces avantages pourraient inclure :

- l'harmonisation de la contribution de l'équipe de développeurs ;
- une amélioration de la planification ; surtout des dates limites et des durées ;
- une réduction du risque de dépassement des budgets et des risques d'échec ;
- une réduction du risque d'omission d'un besoin utilisateur ;
- l'amélioration de la crédibilité et du prestige du département informatique ;
- la possibilité de démarrer la formation des utilisateurs plus tôt ;
- un encouragement à la création de systèmes faciles à installer.

(1) Traduit de l'anglais par les auteurs.

4. SYNTHÈSE ET DISCUSSION

Dans cette section, nous proposons une synthèse des difficultés résultant de l'écart évident entre les besoins des développements informatiques et les méthodes disponibles. Une analogie avec la blague du fêtard et du lampadaire(2) a été utilisée par plusieurs auteurs pour mettre en évidence les problèmes qui existent en termes de méthodologies (Angell & Smithson, 1991 ; Baskerville *et* al., 1992 ; Boehm, 1988 ; Fitzgerald, 1994). Cette histoire permet d'illustrer certains des problèmes essentiels qui se posent dans l'utilisation des SDM traditionnelles. Finalement, nous présentons des arguments en faveur d'un nouveau paradigme qui retourne aux sources des problèmes du développement informatique et qui tient compte des nouvelles technologies et des nouvelles formes organisationnelles émergentes.

4.1. Regards théoriques sur la méthodologie

Etant donné le caractère peu concluant des recherches qui ont essayé de comparer et d'évaluer les méthodologies de développement informatique, il est peut-être juste de se demander si nous, chercheurs dans ce domaine, n'avons pas imité le fêtard de l'histoire : avons-nous cherché à conduire des recherches sur les méthodologies existantes parce que cela semblait plus facile, au lieu de réfléchir sur les problèmes réels du développement informatique ? L'étude de ces problèmes est, bien sûr, bien plus complexe car elle touche à des aspects politiques et organisationnels bien plus difficiles à cerner qu'une simple analyse des activités des acteurs impliqués dans le développement de systèmes sur la base de méthodologies précises et bien définies. Il n'en reste pas moins que de telles études, même difficiles à mettre en œuvre, pourraient apporter des solutions bien plus crédibles et efficaces aux problèmes bien réels que de nombreuses études ont révelés.

4.2. Regards pratiques sur la méthodologie

En ce qui concerne la pratique du développement des systèmes, on peut légitimement se demander si les développeurs en entreprises n'utilisent pas les méthodologies plutot comme support moral que comme guide. Ainsi, les entreprises continuent à utiliser des méthodes peu efficaces et peu utiles pour donner l'impression que tout est fait dans le respect des règles. Dans certains cas, ces méthodes sont utilisées par des entreprises de conseil en informatique et des entreprises de développement de logiciels pour impressionner leurs clients et leur prouver qu'ils ont vraiment affaire à des professionnels. Dans d'autres cas, l'adoption d'une SDM commerciale est justifiée par la nécessité d'obtenir la certification ISO. Quand les méthodes traditionnelles sont utilisées dans ces

(2) Cette blague raconte l'histoire d'un homme qui, rentrant chez lui tard et pris de boisson, avait perdu sa montre. Il la cherchait à la lumière d'un lampadaire quand un passant lui proposa son aide. Voulant savoir exactement où la montre avait été perdue, il demanda au fêtard qui lui expliqua qu'il l'avait perdue plus loin dans la rue, mais qu'il était plus facile de chercher là où la lumière était la meilleure.

contextes, les entreprises courent le risque de voir certains programmeurs, surtout les moins expérimentés, suivre aveuglément les étapes de leur méthode au détriment de l'avancement du projet. Dans ce cas, l'importante flexibilité et l'intelligence naturelle des développeurs sont perdues au profit de l'application automatique de la méthode.

L'importance de la question des performances des informaticiens reste une question vitale pour notre discipline. De nombreux auteurs ont identifié des problèmes très fréquents qui se traduisent par une "crise du logiciel" dont on perçoit les effets dans certains exemples d'échecs informatiques notoires qui ont des échos jusque dans la presse grand public. Nous pensons que ces difficultés sont le produit du retard des méthodes disponibles à l'heure actuelle par rapport aux besoins des entreprises. La création de nouvelles méthodologies plus adaptées aux nouvelles formes organisationnelles et à la complexité croissante des applications qui doivent être developpées pourrait apporter des solutions pour résoudre cette crise. Ce point de vue a déjà été mis en avant par Larrasquet et Clare (1996) qui ont souligné à quel point il était important d'essayer d'exploiter le potentiel des technologies nouvelles (particulièrement en termes de communication) pour le bénéfice du développement des systèmes informatiques.

Le nouvel environnement économique exige que les entreprises agissent plus vite et à plus court terme. Le développement doit devenir plus rapide et la livraison des systèmes doit être accélérée par rapport à ce qui est possible avec les méthodes traditionnelles. En fait, il est même possible que ces dernières soient en partie responsables des problèmes identifiés dans l'étude présentée dans cet article. Ce qui est certain, c'est que l'application automatique de méthodologies génériques ne suffira pas à atteindre les objectifs d'informatisation des entreprises et d'accroissement de l'autonomie des employés.

5. CONCLUSIONS

Plus de recherches sont donc nécessaires pour établir la nature de l'environnement dans lequel les systèmes sont aujourd'hui développés et les besoins réels des entreprises à travers l'étude de cas réels. Dans la plupart des cas, les informaticiens ont développé des techniques personnelles qui correspondent à leurs besoins et ils refusent d'utiliser les méthodes traditionnelles pour d'excellentes raisons basées sur leur expérience et non par ignorance comme on l'a parfois écrit. L'étude de ces pratiques personnelles dans un grand nombre de projets de taille et de complexité différentes pourra livrer les fondations des méthodologies nouvelles. Ces méthodologies devront non pas révolutionner la pratique du développement informatique, mais fourniront une base conceptuelle plus adaptée à la réalité des projets informatiques du vingt et unième siècle.

Avec l'aide de ces nouvelles méthodes, le *développeur des temps modernes* devra utiliser une combinaison de technologies anciennes et nouvelles de façon innovatrice pour l'aider à structurer ses connaissances, travailler mieux en groupe et développer plus vite. En particulier, la messagerie électronique et le group-

ware devront être exploités pour un meilleur partage des connaissances et pour faciliter l'évolution rapide de la structure et de la composition des équipes de développement. Ainsi, les développeurs les plus expérimentés pourront se "relayer" à la tête des équipes de développement en fonction des étapes du projet concerné et de leur compétence spécifique. Finalement, il faudra trouver le moyen d'améliorer l'apprentissage (le "learning") à tous les stades du projet au lieu de le cantonner à la fin du projet dans une hypothétique phase d'audit ou de revue nécessairement plus ou moins bâclée ou le plus souvent remplacée par une phase intense de "documentation". Ces initiatives devront contribuer à une accélération du processus de développement, accélération qui est vitale pour la satisfaction des besoins actuels des entreprises.

BIBLIOGRAPHIE

Adam, F. et Cahen, F.D. (1997), « Socrate, or when organisations try to purchase new systems instead of developing them », *ESRC Research and Discussion Papers*, ref. 97/10, December 1997.

Agresti, W. (1986), *New Paradigms for Software Development*. IEEE Computer Society Press, Washington DC.

Ahituv, N., Hadass, M. et Neumann, S. (1984), « A flexible approach to information system development ». *MIS Quarterly*, Vol. 8, n° 2, p. 69-78.

Angell, I. et Smithson, S. (1991), *Information Systems Management : Opportunities and Risks*, MacMillan, London.

Avison, David and Fitzgerald, Guy (1995), *Information Systems Development : Methodologies, Techniques and Tools*, Information Systems Series, McGraw Hill, London.

Bansler, J. et Havn, E. (1994), « Information systems development with generic systems », in Baets, W. (Ed), *Proceedings of Second European Conference on Information Systems*, Nijenrode University Press, Breukelen, p. 707-718.

Baskerville, R., Travis, J. et Truex, D. (1992), « Systems without method : the impact of new technologies on information systems development projects ». In Kendall, K., DeGross, J. and Lyytinen, K. (eds.) *The Impact of Computer Supported Technologies on Information Systems Development*, Elsevier Science Publishers B.V., North Holland Press, p. 241-269.

Bessagnet, M.N., Larrasquet, J.M. et Jayaratna, N. (1994), « Object Oriented approaches in the analysis and design of information systems to support modern organisational forms », In Lissoni, Richardson, Miles, Wood-Harper and Jayaratna (eds) *Proceedings of 2nd Annual Conference on IS Methodologies*, Heriot-Watt University, p. 177-184.

Boehm, B. (1988), « A spiral model of software development and maintenance ». *IEEE Computer*, Vol. 21, n° 5, p. 61-72.

Bohm, C. et Jacopini, G. (1966), « Flow diagrams, Turing machines and languages with only two formation rules ». *Communications of the ACM*, May, p. 366-371.

Brown, P. (1985), « Managing software development ». *Datamation*, April 15, p. 133-136.

Chen, P. (1976), « The entity relationship model-towards a unified view of data », *ACM Transactions on Database Systems*, Vol. 1, n° 1, p. 9-36.

Cox, B. (1990), « Planning the software industrial revolution ». *IEEE Software*, November, p. 25-33.

Davis, A., Bersoff, E. et Comer, E. (1988), « A strategy for comparing alternative software development life cycle models ». *IEEE Transactions on Software Engineering*, October, p. 1453-1460.

DeGrace, P. et Stahl, L. (1990), *Wicked Problems, Righteous Solutions : A Catalogue of Modern Software Engineering Paradigms*. Yourdon Press,

Prentice Hall, Englewood Cliffs, New Jersey.

DeMarco, T. (1978), *Structured Analysis and System Specification*, Yourdon Press, New Jersey.

Downs, E., Clare, P. et Coe, I. (1992), *Structured Systems Analysis and Design Method : Application and Context*. Prentice-Hall International (UK), Hertfordshire.

Dyke, R. et Kunz, J. (1989), « Object-oriented programming ». *IBM Systems Journal*, Vol. 28, n° 3.

Enger, N. (1981), « Classical and structured system life-cycle phases and documentation », in Cotterman, W., Couger, J., Enger, N. and Harold, F. (Eds) (1981) *Systems Analysis and Design : A Foundation for the 1980s*, Elsevier, New York, p. 1-24.

Firesmith, D. (1993), *Object-Oriented Requirements Analysis and Logical Design*, Wiley and Sons, New York.

Firesmith, D. (1992), « Take a flying leap : the plunge into object-oriented technology ». *American Programmer*, Vol. 5, n° 8, p. 17-27.

Fitzgerald, Brian (1997), « The use of systems development methodologies in practice : a field study », *Information Systems Journal*, Vol. 7, p. 201-212.

Fitzgerald, Brian (1996), « An investigation of the use of systems development methodologies in practice », in Coelho, J. et al. (Eds) *Proceedings of the 4th European Conference on Information Systems*, Lisbon, p. 143-162.

Fitzgerald, Brian (1994), « Whither systems development : time to move the lamppost ? », in Lissoni *et al.*, (eds), *Proceedings of Second Conference on Information Systems Methodologies*, p. 371-380.

Folkes, S. et Stubenvoll, S. (1992), *Accelerated Systems Development*, Prentice Hall, London.

Friedman, A. (1989), *Computer Systems Development : History, Organisation and Implementation*, Wiley & Sons, Chichester.

Gilb, T. (1988), *Principles of Software Engineering Management*, Addison Wesley, UK.

Guimaraes, T. (1985), « A study of application program development techniques ». *Communications of the ACM*, May, p. 494-499.

Hardy, C., Thompson, B. and Edwards, H. (1995), « Problems associated with the customisation of structured methods », in Jayaratna, Miles, Merali and Probert (Eds.) *Proceedings of the Third Conference on Information Systems Methodologies*, p. 211-220.

Hough, D. (1993), « Rapid delivery : an evolutionary approach for application development », *IBM Systems Journal*, Vol. 32, n° 3, p. 397-419.

Jenkins, A., Naumann, J. et Wetherbe, J. (1984), « Empirical investigation of systems development practices and results ». *Information & Management*, Vol. 7, p. 73-82.

Larrasquet, J.M. et Clare, P. (1996), « The implication of new organisational forms and emerging technology for IS methodologies ». In Jayaratna, N. and Fitzgerald, B. (eds) *Proceedings of 4th Annual Conference on IS Methodologies*, University College Cork, p. 407-413.

Mahmood, M. (1987), « Systems development methods a comparative investigation », *MIS Quarterly*, p. 293-311.

Nilsson, A. (1990), « IS development in an application package environment », In Wrycza, S (Ed) *Proceedings of Second International Conference on IS Developers Workbench*, University of Gdansk, p. 444-466.

O'Duffy, M. (1992), *Centre for Software Engineering Newsletter*, p. 2-3.

Orr, K. (1989), « Methodology : the experts speak ». *BYTE*, April, p. 221-233.

Palvia, P. and Nosek, J. (1993), « A field examination of system life cycle techniques and methodologies », *Information and Management*, n° 25, p. 73-84.

Parnas, D. (1972), « On the criteria to be used in decomposing systems into modules ». *Communications of the ACM*, Vol. 15, n° 12, p. 1053-1058.

Rockart, J. et De Long, D. (1988), *Executive Support Systems*, Dow Jones-Irwin, Homewood, Illinois.

Royce, W. (1970), « Managing the development of large software systems ». *Proceedings of IEEE Wescon.*

Stevens, W., Myers, G. et Constantine, L. (1974), « Structured design ». *IBM Systems Journal*, Vol. 13, n° 2, p. 115-139.

Sumner, M. et Sitek, J. (1986), « Are structured methods for systems analysis and design being used ? » *Journal of Systems Management*, June, p. 18-23.

Takeuchi, T. et Nonaka, I. (1986), « The new product development game », *Harvard Business Review*, January-February.

Thomann, J. (1994), « Data modelling in an OO world ». *American Programmer*, Vol. 7, n° 10, p. 44-53.

Topper, A. (1992), « Building a case for object-oriented development », *American Programmer*, Vol. 5, n° 8, p. 36-47.

Verity, J. (1987), « The OOPS revolution ». *Datamation*, May 1, p. 73-78.

Ward, P. (1991), « The evolution of structured analysis : Part I : the early years ». *American Programmer*, Vol. 4, n° 11, p. 4-16.

Yourdon, E. (1991), « Sayonara, once again, structured stuff ». *American Programmer*, Vol. 4, n° 8, p. 31-38.