CHAPITRE III

LE LOGICIEL :

UN PRODUIT ET UNE VALEUR "INSAISISSABLES" ?

Le logiciel apparaît par bien des aspects comme un produit particulièrement difficile à appréhender. Il l'est tout d'abord en raison de son extraordinaire diversité fonctionnelle. Les logiciels pénètrent les domaines d'activité les plus variés et sont confrontés à des problèmes de plus en plus complexes, auxquels il semble que l'informatisation, grâce à l'augmentation de la puissance des matériels et à la baisse de leur coût, puisse apporter une solution. En même temps, dans certains cas l'informatisation d'une situation peut contribuer à la complexifier davantage, ce qui nous amènera à nous interroger sur les significations multiples de la notion de complexité. Dans l'économie des logiciels se produisent de multiples innovations qui concernent l'ensemble des dimensions de la production des logiciels : apparition de nouveaux produits, de nouvelles méthodes de production, utilisation de nouveaux outils. Si parfois les changements peuvent être extrêmement rapides (par exemple un logiciel de navigation sur Internet est devenu un outil banal et largement utilisé alors qu'il n'existait pas il y a quelques années), il faut également prendre en compte le temps et les efforts d'assimilation et d'apprentissage, qui expliquent que la diffusion de certaines innovations puisse être beaucoup plus lente. Ces évolutions qui se produisent sur une période très courte (la production de logiciel comme une activité distincte n'a pas plus de trente années d'existence) expliquent l'instabilité technologique, qui semble caractéristique de l'économie du logiciel et qui rende plus difficile son analyse.

L'analyse n'est pas plus facile sur un plan économique. Qu'est-ce économiquement qu'un logiciel ? S'agit-il d'un bien ou d'un service ? Est-ce un produit sur lequel peuvent s'exercer des droits de propriété ? A quelles conditions techniques et/ou juridiques ? Qu'est-ce qui détermine son prix ? Les théories de la valeur des économistes peuvent-elles lui être appliquées ?

Pour répondre à ces questions, nous verrons comment les spécificités technico-économiques des logiciels rendent difficile l'utilisation des catégories habituelles des économistes (section II). Auparavant, nous étudierons les grandes lignes de l'évolution technologique extrêmement rapide, caractéristique de l'économie du logiciel (section I). La séparation de "l'insaisissabilité" des logiciels en une dimension plus technique et une dimension plus théorique a été opérée pour faciliter l'exposition des raisonnements, ces deux dimensions étant intimement liées dans la réalité.

Section I - L’ « insaisissabilité » technique : l'évolution extrêmement rapide des produits, de leurs domaines d'utilisation et des techniques de production

L'économie du logiciel connaît des changements permanents qui correspondent à l'ensemble des modèles d'innovations mis en évidence pour l'ensemble de l'économie (C). Ces innovations sont une réponse à la complexité croissante des problèmes que doivent résoudre les logiciels (B), sur une période relativement brève, l'économie du logiciel ne s'étant constituée comme activité économiquement séparée que depuis une trentaine d'années (A).

A - LA JEUNESSE DE LA PRODUCTION DE LOGICIELS COMME ACTIVITE SEPAREE

Si la production d'ordinateurs a environ un demi-siècle d'existence (elle commence après la seconde guerre mondiale), la production de logiciels comme une activité séparée ne date que du début des années soixante-dix[1]. Certes, l'utilisation des ordinateurs a toujours nécessité l'existence d'une activité de programmation[2], mais celle-ci était exclusivement effectuée par les utilisateurs, les premiers ordinateurs étant fournis "nus". Au début des années soixante, les matériels étaient livrés avec des logiciels (principalement des systèmes

[1] "L'informatique est une industrie née il y a cinquante ans, et pourtant la question du logiciel n'a pas été posée pendant les vingt-cinq premières années, bien que celles-ci aient été caractérisées par une immense créativité logicielle" (Christian Genthon, Denis Phan, 1999, p. 172).

[2] Le premier livre écrit sur la programmation, à partir des travaux de l'équipe de l'EDSAC, date de 1951 (Jean-Yvon Birrien, 1990, p. 53).

d'exploitation et des langages de programmation) mais seul le matériel faisait l'objet d'une facturation.

L'émancipation économique du logiciel par rapport au matériel a résulté de plusieurs phénomènes :

- Au milieu des années soixante, IBM décide de produire une série d'ordinateurs (les 360) compatibles entre eux et devant couvrir l'ensemble des besoins. La mise en application de cette décision qui aura un impact décisif sur le développement de l'informatique traditionnelle et son contrôle par IBM (cf. chapitre II), connaît de sérieuses difficultés dans sa partie logicielle : retards dans les développements, nécessité de fournir des services importants d'assistance et de formation pour résoudre les problèmes rencontrés par les utilisateurs (Gérard Dréan, 1996 A, p. 31). De simple produit complémentaire à la vente de matériel, le logiciel devient "un poste de dépenses majeur et un facteur concurrentiel souvent déterminant pour le constructeur" (idem, p. 19). D'autre part, la non-facturation des services de plus en plus substantiels pose des problèmes nouveaux, comme la question de savoir si IBM devait continuer à assurer la maintenance des matériels IBM fournis par des revendeurs d'occasion qui prenaient de plus en plus d'importance.

- Dans le même temps, en réponse à la demande de logiciels applicatifs et de services de plus en plus diversifiés, se développent des Sociétés de Service et de Conseil en Informatique (S.S.C.I.), qui, si elles facturent leurs prestations, présentent l'avantage d'être (relativement) indépendantes des constructeurs et de disposer de compétences spécifiques, dans une période de pénurie de personnels qualifiés (Eric Labat, 1984, p. 195).

- Enfin, IBM se trouve sous la pression des procédures judiciaires au titre des lois anti-trust.

C'est dans ce contexte qu'IBM adopte en 1969 une politique d'*unbundling* ou facturation séparée des matériels, des logiciels, de l'assistance technique et de la formation, en anticipant un mouvement qui paraissait inéluctable (Jean-Michel Treille, 1973, p. 97). Dans un premier temps seule une partie des services et du logiciel devient facturable, l'autre partie restant incluse dans le prix du matériel. Pour les logiciels seuls les logiciels applicatifs non essentiels à l'exploitation (*Program Products*) sont facturés séparément, les logiciels "nécessaires à l'exploitation" (*Systems Control Programs*) restant compris dans le prix du matériel. Cette

disposition ne sera que transitoire et à partir des années quatre-vingt, tous les logiciels fournis par IBM sont facturés. La décision *d'unbundling* marque la naissance d'une économie du logiciel indépendante, en permettant notamment l'essor des SSCI. Ce mouvement s'amplifiera avec le développement des progiciels à partir des années quatre-vingt. Il est intéressant de noter que si certaines évolutions récentes semblent prendre la direction inverse en proposant des solutions informatiques globales, intégrant des composants matériels et logiciels, elles ne remettent pas en cause l'autonomie de l'économie du logiciel, les fournisseurs de ce type de solutions achetant la plus grande part des logiciels qu'ils intègrent.

C'est donc sur une période très courte (trente ans) qu'il faut analyser des évolutions correspondant schématiquement au passage de l'artisanat à la grande industrie automatisée, un passage qui a pris plusieurs siècles dans d'autres secteurs. En se référant aux trois grandes étapes de l'évolution professionnelle dans l'industrie mises en évidence par Alain Touraine (1962), on peut dire que la production des logiciels est passée rapidement de l'étape A (prédominance de l'action autonome de l'ouvrier qualifié, en l'occurrence l'informaticien) à l'étape B (prédominance de l'organisation centralisée du travail intégrant des analystes, des programmeurs…), et qu'elle entre déjà dans l'étape C de l'automatisation de la fabrication avec le développement d'outils de génie logiciel. En même temps la nature des problèmes que doivent résoudre les logiciels a considérablement évolué.

B - LA COMPLEXITE CROISSANTE DES PROBLEMES A RESOUDRE

Un large accord existe sur la complexité croissante des problèmes que doivent résoudre les logiciels, et l'accroissement de la complexité des logiciels qui en résulte[3]. Le logiciel est plus complexe et plus difficile à maîtriser que le matériel[4]. La critique, fréquente, de cette situation, omet le fait que c'est dans la nature même du logiciel de réaliser ce qui est trop complexe pour le matériel (Gérard Dréan, 1996 A, p. 198). Une idée nouvelle ou l'exploration d'un nouveau domaine se traduit d'abord par le développement de programmes. Ce n'est que

[3] Par exemple, Patrick Jaulent estime que "l'un des traits les plus marquants de ces dernières années en matière de logiciel est l'accroissement de la complexité des applications réalisées" (1992, p. 18). Frederick P. Brooks considère que "les entités logicielles sont sans doute les constructions les plus complexes réalisées par l'homme" (1996, p. 158).

[4] "La description externe d'un système logiciel est 10 à 20 fois plus longue que celle de l'ordinateur proprement dit" (Frederick P. Brooks, 1996, p. 35).

lorsque la complexité de cette nouvelle application aura été maîtrisée qu'il deviendra éventuellement possible et souhaitable d'inscrire certaines des fonctions nécessaires sur des composants électroniques spécialisés.

Au-delà de ce constat, il est nécessaire de définir précisément ce qu'est la complexité (1), avant de voir que la complexité croissante des logiciels est tout à la fois le produit d'un environnement informatique de plus en plus complexe (2) et de la volonté d'automatiser des fonctions de plus en plus complexes (3). Enfin si les logiciels doivent intégrer cette complexité croissante, il faut également tenir compte qu'en retour eux-mêmes peuvent être créateurs de complexité (4).

1 - Les significations multiples de la notion de complexité

La complexité n'est pas simple à définir… La complexité peut prendre plusieurs significations différentes mais liées. La complexité peut évoquer les difficultés pour résoudre un problème (considéré comme complexe). Cette complexité que Faïz Gallouj (1994, p. 196) qualifie de *complexité-compréhension*, est relative aux compétences de la personne ou de l'organisation.

La complexité peut renvoyer à l'incertitude, qui résulte de la méconnaissance et/ou du caractère imprévisible de certains phénomènes, et qui empêche d'avoir une bonne vision du comportement d'un système qualifié de complexe. La fréquence et l'importance des changements peuvent accroître cette complexité, désignée comme étant une *complexité-mouvement* (Faïz Gallouj, 1994, p. 196).

En privilégiant une dimension quantitative, la théorie de l'information définit la complexité comme une quantité d'information, liée au nombre et à la variété des éléments d'un système[5], dont la combinatoire détermine le nombre d'états qu'il peut prendre[6]. En ce sens un

[5] Or, normalement, un logiciel bien construit n'a pas deux parties identiques (du moins au-dessus du niveau de l'instruction). Quand il existe deux parties identiques, elles sont fusionnées en une seule sous forme d'un sous-programme qui sera appelé plusieurs fois. De ce point de vue, un logiciel est différent d'un ordinateur, d'un bâtiment ou d'une automobile qui comprennent beaucoup d'éléments en de multiples exemplaires (Frederick P. Brooks, 1996, p. 158).

[6] Les ordinateurs sont déjà d'un niveau de complexité extrêmement élevé car ils ont un très grand nombre d'états. Les systèmes logiciels ont un nombre d'états supérieur de plusieurs ordres de grandeur à celui des ordinateurs (Frederick P. Brooks, 1996, p. 158).

système complexe comporte beaucoup de pièces très différentes, ce qui le rend difficile et coûteux à assembler, et il regroupe une quantité d'information élevée puisque chaque pièce est un cas particulier. Cette vision de la complexité renvoie à la notion de *complexité-volume* de Faïz Gallouj.

Pour l'analyse systémique, ce qui caractérise la complexité c'est moins le nombre de composants différents d'un système que l'importance des interactions entre ces éléments. Dans un système complexe "le tout est plus que la somme des parties" et "étant donné les propriétés des parties et les lois de leurs interactions, l'inférence des propriétés du tout n'est pas une question triviale" (Herbert Simon, 1974, p. 106-107). C'est ce qui explique que pour construire l'intelligibilité d'un système complexe, il soit nécessaire de le modéliser (processus de compréhension), alors que l'intelligibilité d'un système compliqué peut être découverte par simplification (processus d'explication) (Jean-Louis Le Moigne, 1990, p. 11). Jean-Louis Le Moigne (1977) définit neuf niveaux de complexité, classés dans l'ordre de complexité croissante des objets :

Niveau 1 : L'objet est passif.

Niveau 2 : L'objet est actif. Il transforme des flux de façon prédéterminée (processeur).

Niveau 3 : L'objet actif comporte un processeur de régulation.

Niveau 4 : L'objet est capable de capter de l'information et de l'utiliser dans son fonctionnement.

Niveau 5 : L'objet devient capable de décisions. Il possède un processeur décisionnel autonome.

Niveau 6 : L'objet possède une mémoire qui permet à son processeur actif et à son processeur décisionnel de fonctionner en tenant compte non seulement du comportement actuel mais également des comportements passés.

Niveau 7 : Chacun des processeurs (actif, décisionnel, mémoire) est lui-même un système complexe. Ils sont appelés respectivement Système Opérant, Système de Pilotage, Système d'Information et leur coordination est indispensable.

Niveau 8 : L'objet est capable d'auto-organisation, d'où la nécessité de la présence d'un système d'imagination-conception dans le système de pilotage.

Niveau 9 : L'objet s'autofinalise (présence d'un système de finalisation). L'objet peut définir ses objectifs et les faire évoluer.

Enfin, les informaticiens utilisent la notion de *complexité algorithmique* qui étudie la relation entre la taille d'un problème, et le volume de mémoire et/ou le nombre de pas de calcul qui sera nécessaire pour le résoudre. Cette complexité algorithmique peut être appréhendée de deux façons différentes. La première est une mesure de la performance des systèmes qui prend en compte les contingences des machines réelles. La seconde, de nature

plus théorique et d'essence mathématique, est une fonction qui détermine le nombre d'instructions élémentaires abstraites nécessaires pour exécuter l'algorithme en fonction du nombre d'éléments qui caractérisent les données en entrée de l'algorithme. En pratique, on ne s'intéresse qu'au terme d'ordre le plus élevé et on néglige les facteurs constants. Par exemple, un algorithme de tri dont la complexité est en $O(n^2)$ nécessite un ordre de grandeur de n^2 opérations pour trier n éléments. Les meilleurs algorithmes de tri sont de complexité O(n Log n) ; ils sont préférables au précédent puisque quand n (nombre d'éléments à trier) augmente, n Log n (nombre d'instructions à exécuter) croît moins rapidement que n^2. Il existe deux grandes classes d'algorithmes : ceux dont la complexité s'exprime sous la forme d'un polynôme et ceux dont la complexité ne peut être bornée par un polynôme (NP *Non Polynomial*, car contenant des exponentielles). Dans ce dernier cas, l'augmentation du nombre d'éléments en entrée sature très rapidement l'ordinateur le plus puissant et l'algorithme est en pratique inutilisable. Les problèmes de nature NP sont très fréquents[7] (notamment les problèmes d'optimisation) et peuvent avoir une apparence anodine : l'exemple le plus célèbre est celui du voyageur de commerce qui veut minimiser son trajet pour parcourir n villes. Dans ces situations l'informatisation du problème nécessite de recourir à des heuristiques. A la différence des algorithmes dont on peut démontrer qu'ils donnent la solution optimale, les heuristiques sont des raisonnements formalisés (en général moins coûteux en nombre d'instructions) dont on tient pour plausible qu'ils donneront une solution satisfaisante (*satisfactum* ou *satisficing*, Herbert Simon, 1974, p. 83) du problème.

En conclusion, quel que soit le sens que l'on attribue à la notion de complexité, il apparaît que les logiciels doivent répondre à des problèmes de complexité croissante, ce que permet l'amélioration des performances du matériel.

2 - Amélioration des performances du matériel et environnement informatique de plus en plus complexe

C'est en effet, l'augmentation exponentielle de la puissance du matériel qui permet l'utilisation de logiciels de plus en plus complexes. Sa meilleure illustration est la "loi" de Gordon Moore (cofondateur d'Intel) énoncée à la fin des années soixante, selon laquelle la

[7] "Par malchance pour les ordinateurs, les algorithmes intéressants pou les humains ont une fâcheuse tendance à avoir une complexité élevée (…). Pour ce type de situation, il peut ne pas exister d'algorithmes, soit

puissance des microprocesseurs et la capacité des mémoires doubleraient tous les deux ans (révisée ensuite en tous les 18 mois) : trente ans de vérification de cette loi équivalent à un rapport prix/performances multiplié par un million[8].

La complémentarité économique entre les producteurs de matériels et de logiciels est forte, ce que traduit l'expression "Wintel" (Windows + Intel) qui désigne l'alliance objective existant dans la micro-informatique entre les intérêts d'Intel pour les microprocesseurs et de Microsoft pour les logiciels. L'augmentation de la puissance des matériels suscite la production de logiciels plus "gourmands" en ressources matérielles. En retour, le développement de ces logiciels incite les utilisateurs à renouveler leurs matériels.

Le rapport dynamique qui lit le matériel et le logiciel est également un rapport technique concernant leur production. Ce sont les progrès du matériel informatique qui ont, entre autres facteurs, permis de développer des logiciels plus complexes. Réciproquement ce sont les progrès dans les logiciels notamment de CAO et de DAO qui ont permis de produire ces composants de plus en plus puissants (Jean-Louis Caccomo, 1996, p. 31).

La complexité croissante des systèmes informatiques résulte aussi de la diversité grandissante des multiples périphériques d'entrée et de sortie d'informations qui ont vu le jour. Ici également, les interdépendances sont fortes. C'est l'amélioration du rapport prix/performance des composants matériels de base et les perfectionnements des logiciels qui ont rendu techniquement possibles et économiquement viables l'utilisation de ces périphériques, qui facilitent les communications avec un système informatique. Mais l'emploi de ces périphériques nécessite des ressources matérielles supplémentaires et complexifie le développement des logiciels qui les pilotent.

---

parce que l'algorithme est inconnu, soit parce que l'algorithme requiert des ressources de calcul impossibles à réunir" (Jacques Printz, 1998, p. 29-33).

[8] Michel Catinat souligne qu'il "n'existe pas d'autre exemple où le progrès technologique est comparable en intensité et en durée". Les performances des technologies informatiques, à savoir une vitesse de traitement et une capacité de stockage multipliées par 100 tous les 10 ans depuis les années soixante, donneraient dans le transport par rail les résultats suivants : la durée d'un trajet Paris Marseille serait passée de dix heures dans la décennie soixante, à six minutes dans la décennie soixante-dix, à quatre secondes dans la décennie quatre-vingt, et à quatre centièmes de seconde dans la décennie quatre-vingt-dix, avec une capacité de transport multipliée par un million sur l'ensemble de la période. Les progrès pourtant spectaculaires que constituent l'apparition du TGV apparaissent minimes par comparaison (1998, p 38).

Une cause supplémentaire de complexité est l'interconnexion de plus en plus grande des différents matériels informatiques : dans certains systèmes d'informations, le nombre d'équipements connectés est fréquemment de l'ordre du millier (en 1997, le système de réservation Amadeus comptait 180 000 équipements répartis dans plus de 100 pays). Cette interconnexion a dépassé le cadre des systèmes d'informations particuliers pour concerner tendanciellement l'ensemble des systèmes informatiques existants avec la naissance et le développement d'Internet. Elle n'a pu s'effectuer que grâce à une transformation architecturale majeure des systèmes informatiques : d'une conception hiérarchique, appelée "maître-esclave", où une machine (gros système ou mainframe qui concentre l'essentiel des ressources informatiques), supervisait les actions des autres machines (principalement des terminaux "passifs"), on est passé à une architecture "client-serveur", où toute machine peut selon les situations être traitée en client ou devenir serveur, c'est à dire fournisseur d'un service ou d'une ressource. Le terme serveur fait référence à tout processus qui reçoit une demande de service venant d'un client via un réseau, traite cette demande et renvoie le résultat au demandeur (client). Ce type d'architecture "distribuée" permet le traitement coopératif d'applications, c'est à dire la communication directe de deux applications via un réseau, ou encore la communication directe de deux processus d'une même application répartie sur un réseau, chaque processus s'exécutant sur la machine la plus appropriée. Le modèle client-serveur a permis une rationalisation de l'utilisation des ressources matérielles, en exploitant l'augmentation de la puissance à un coût réduit représentée par les micro-ordinateurs et les stations de travail. Par contre, il a nécessité d'importants efforts en termes de développement des logiciels pour traiter les problèmes de communication entre systèmes, de gestion de bases de données distribuées et de gestion de systèmes distribués, et une refonte des applications existantes, voire de toute l'organisation informatique[9]. La souplesse d'évolution d'une telle architecture, et en conséquence la variété croissante des services proposés, expliquent la rapidité de sa diffusion : aux Etats-Unis le pourcentage des applications sur systèmes client-serveur est passé de 27 % en 1993 à 51 % en 1995 et est estimé à 76 % en 1997 (OCDE, 1997 B, p. 35).

[9] Une architecture client-seveur réduit les dépenses en matériel mais augmente celles en logiciels et en services plus complexes (Eurostaf, 1996 A, p. 87).

3 - L'objectif d'automatisation de fonctions de plus en plus complexes

Mis à part le calcul scientifique, l'informatisation a concerné au départ des tâches de *back-offic*e simples et répétitives (gestion de la paye, facturation, opérations bancaires élémentaires…). Si le volume des opérations pouvait être important, ce qui justifiait l'utilisation de l'ordinateur, les logiciels étaient peu complexes (ce qui ne signifie pas que rapportées aux connaissances de l'époque ils aient été simples à mettre au point), d'autant que les tâches à informatiser avaient souvent été préalablement standardisées et taylorisées et que leur traitement informatique s'effectuait en temps différé.

L'intégration des différentes fonctions

Une première dimension de la complexification des logiciels provient de deux phénomènes liés : la volonté d'intégrer les différentes procédures – elles-mêmes de complexité croissante - qui avaient été automatisées séparément (gestion de la production, des finances et de la logistique), et d'effectuer ces différentes opérations de façon interactive et en temps réel. Cette volonté d'une informatisation complète du système de gestion de l'entreprise correspond à une évolution de la vision du système d'information de l'entreprise comme étant au cœur de celle-ci. Elle s'est traduit par le développement du marché des logiciels de gestion globale d'entreprise (ERP, *Entreprise Ressources Planning*), des bases de données, de la gestion des réseaux. Cette globalisation de l'informatisation peut avoir en retour des conséquences importantes sur le fonctionnement de l'entreprise : "il ne s'agit plus seulement d'« informatiser » un service ou une activité, mais bien de repenser les modes de production et d'organisation d'une activité, compte tenu des potentialités de l'informatique" (Jacques De Bandt, 1995, p. 84). Les technologies de l'information ne servent plus uniquement à automatiser des tâches et des métiers existants dans l'entreprise, et sont de plus en plus utilisées pour repérer et faciliter l'émergence de nouveaux métiers et services.

Les types de décisions informatisées

La deuxième dimension de la complexification qui s'est opérée en parallèle avec la précédente concerne le type de décisions modélisées dans les logiciels. Les décisions peuvent être classées en trois niveaux : opérationnel, coordination, stratégique (François Pichault, 1990, p. 73). Certes tout processus de décision peut se décomposer en phases d'identification, d'évaluation, de choix et de ratification (Herbert Simon, 1980) et comprend des périodes

d'investigation, de transmission et de traitement d'information, mais le type d'informations concernées est différent.

Par exemple, Takayasu Miyakawa (1997, p. 67) distingue quatre modalités d'usage de l'information dans la sphère productive : l'information routinière, l'information mercatique (développement et protection des marchés), l'information opérationnelle (liée à la conception et à la mise en œuvre de la production), l'information managériale (caractère stratégique). De même, Luc Rubiello (1997, p. 76-78) différencie l'information de production, l'information tactique et l'information stratégique. L'information de production (ou opérationnelle) manipule des données de base (brutes) volumineuses en taille et en nombre, mais concerne des opérations dont l'informatisation est ancienne à partir d'un outil de base commun - le système de gestion de bases de données (SGBD) - qui sont relativement stables et bien maîtrisées (comptabilité, gestion des stocks, paye, facturation). L'information tactique est obtenue par le traitement des données du niveau précédent et repose sur des outils de pilotage (interrogations avec les langages de quatrième génération L4G, systèmes informatisés d'aide à la décision SIAD). L'information stratégique repose sur des informations issues de la couche tactique, elle correspond à des besoins mal définis et changeant rapidement et s'appuie sur des outils plus récents comme les *Executive Information System* (EIS). Les problèmes décisionnels correspondants sont le contrôle opérationnel (par exemple, le contrôle des stocks) pour l'information de production, le contrôle tactique (par exemple, l'analyse budgétaire) pour l'information tactique, et la planification stratégique (par exemple, la localisation des lieux de production) pour l'information stratégique.

Ces décisions n'ont pas toutes le même niveau de complexité (Eric Brousseau, 1993, p. 224) : certaines peuvent être qualifiées de simples, comme le tri d'une liste par ordre alphabétique, et d'autres de complexes, comme le design d'un nouveau produit. Le premier type de décision est le fruit d'un raisonnement totalement logique (un calcul) et fait partie de ce que Ehud Zuscovitch (1983, p. 51) nomme la partie "algorithmée" de l'entreprise (ou plus généralement d'une organisation). Le deuxième type de décision fait appel à un raisonnement qui n'est pas totalement formalisé (structuré, décomposé en phases analytiques séparées), et/ou repose sur des informations non formalisées. Disposant d'un savoir incomplet sur le monde, la vision du monde des autres agents et l'avenir, le décideur se fie largement à son intuition et à son expérience, fait appel aux résultats de processus d'esssais-erreurs observés dans le passé qu'il remet en perspective, par analogie, avec la situation présente, qui n'est pas

immédiatement déductible d'une situation antérieure et connue (Eric Brousseau, 1993, p. 224). Ces décisions représentent la partie "non algorithmée" de l'organisation (Ehud Zuscovitch, 1983, p. 51). Naturellement les premières décisions qui ont été informatisées correspondent à la partie algorithmée en commençant par les plus simples et les plus fréquentes (routinières). L'informatisation de ces décisions, en permettant la collecte systématique de données facilement manipulables et stockables sur les opérations réalisées, a ouvert des opportunités pour l'automatisation de décisions plus complexes de caractère plus stratégique. On peut citer les exemples de la gestion logistique à partir de la gestion des stocks, de la conception des produits basée sur l'automatisation de leur fabrication. Cette tendance que Ehud Zuscovitch (1983, p. 52) nomme "algorithmisation de la firme", repose sur la distinction (par rapport à l'incertitude caractéristique de la partie non algorithmée) entre une incertitude réductible (difficultés à assembler toute l'information pour prendre une décision dans un temps limité) et une incertitude irréductible (provenant de la simultanéité des actions des autres et des autres événements extérieurs) (idem, p. 51). Si certaines activités de décision échappent à toute tentative de formalisation ou de structuration en raison de "l'irréductibilité de l'être humain" (François Pichault, 1990, p. 13), d'autres ne le sont qu'en raison de l'état actuel de nos connaissances. Les conséquences de cette évolution sont le développement de logiciels plus complexes reposant sur des algorithmes eux-mêmes plus complexes ou sur des heuristiques (outils d'aide à la décision).

On peut caractériser cette évolution en utilisant la typologie des technologies immatérielles de R.L Daft et N.B. Mac Intosh (1978) basée sur deux critères : la variété des tâches d'une part, leur degré de précision, de clarté, d'intelligibilité d'autre part. L'informatisation a d'abord concerné les technologies programmables (variété faible, clarté élevée). Elle a concerné ensuite les technologies de savoir-faire (variété et clarté faibles, par exemple cambiste) et les technologies technico-professionnelles (variété et clarté élevées, par exemple la fonction juridique). Enfin elle atteint les technologies de recherche (variété élevée, clarté faible) dont un exemple est la formulation de la stratégie.

L'automatisation des relations

La troisième dimension de la complexification concerne l'utilisation des technologies de l'information et de la communication pour automatiser des relations ou établir de nouvelles relations entre agents économiques. Jusqu'à la fin des années quatre-vingt, l'application des technologies à des tâches et des fonctions étendait l'automatisation à de nouveaux domaines

d'applications (activités de bureau, technologies de production flexibles) mais sans modification fondamentale par rapport aux phases précédentes de l'automatisation (Alain Rallet, 1997, p. 96). Une rupture qualitative se produit dans les années quatre-vingt-dix avec l'intégration de l'informatique et des télécommunications, qui va permettre aux technologies de l'information et de la communication de réaliser pleinement leur potentiel de "*mediating technology*" (Claudio Ciborra, 1993) en portant sur les mécanismes de coordination entre les agents économiques, sur deux plans complémentaires. Premièrement à l'intérieur de l'organisation avec le développement de nouvelles techniques de travail de groupe (*groupware*) en relation avec des modifications dans l'organisation des activités, plus centrées sur la réalisation complète d'un produit, la maîtrise globale des processus et des événements, et l'accomplissement collectif d'un projet que sur une stricte séparation des tâches. Deuxièmement, à l'extérieur de l'organisation, l'utilisation des technologies de l'information et de la communication pour gérer les relations avec les partenaires extérieurs de l'organisation, que celles-ci soient occasionnelles avec des clients (commerce électronique) ou plus stables avec des fournisseurs et sous-traitants (Echange de Données Informatisées, et plus récemment la mise en place *d'extranet*). Le bon déroulement de ces interactions entre des acteurs différents, disposant d'équipements hétérogènes, concernant des informations textuelles mais aussi sonores et visuelles, avec des exigences croissantes en termes de sécurité, de confidentialité, de fiabilité et de convivialité, nécessite la mise au point de logiciels de plus en plus complexes.

Le traitement des connaissances

La quatrième dimension de la complexification concerne l'utilisation des technologies de l'information pour traiter les connaissances. Après les déboires de l'intelligence artificielle dans sa volonté de produire des connaissances, dont les principales réalisations se sont limitées au développement de systèmes experts avec des succès divers selon les domaines, les tentatives actuelles concernent plus modestement la gestion des connaissances, que celles-ci soient internes à l'organisation (*knowledge management*) ou externes (intelligence économique). Cependant si les objectifs sont moins ambitieux, la complexité (dans tous les sens du terme) des outils logiciels nécessaires à ces activités est par contre extrêmement élevée.

4 - Les logiciels comme créateurs de complexité : la complexité proactive

La plus grande partie de la complexité que doit maîtriser un logiciel provient "des nombreuses institutions humaines et les nombreux systèmes auxquels les interfaces du logiciel doivent se conformer" (Frederick P. Brooks, 1996, p. 159). C'est en effet, le logiciel qui doit s'adapter " parce qu'il est le dernier arrivé sur le terrain ou parce qu'il est perçu comme étant le plus facile à rendre compatible au reste" (idem). Mais les logiciels ne sont pas seulement plus complexes parce qu'ils doivent répondre à des situations de complexité croissante. Dans de nombreux cas, ce sont les potentialités offertes par les logiciels qui contribuent à complexifier (parfois considérablement) les problèmes que devront résoudre les logiciels. La *complexité proactive* désigne cette situation où se crée de la complexité par la production de nouveaux problèmes, au-delà de la complexité inhérente à la résolution des problèmes déjà existants.

Michel Crozier (1963) a montré comment les entreprises évoluent en complexifiant leurs règles. Cette tendance, freinée par les capacités limitées des services organisationnels, peut avec les potentialités de l'informatisation s'exprimer pleinement par l'addition de nouvelles réglementations et procédures de plus en plus sophistiquées. Cette complexité proactive concerne également la multiplication des types de produits, de prestations, de cas différents, de situations qu'il faut envisager, et le raccourcissement des délais de réaction. Le traitement automatique de l'information a facilité le développement des politiques de flux tendus et de zéro stocks qui en retour augmentent la complexité des situations à gérer : par exemple, Districast, distributeur de l'éditeur Casterman, a réussi en optimisant sa logistique, à livrer ses clients en 24 heures. De ce fait, ceux-ci réduisent leurs propres stocks, ce qui a eu pour conséquence une augmentation considérable des commandes d'un montant unitaire plus faible, le volume des ventes restant à peu près stable. En réponse à cette complexité supplémentaire, Districast a automatisé la préparation de ces commandes (Catherine Palierne, 1998, p. 110). Un exemple caricatural de complexité créée en partie artificiellement est fourni par le système Socrate de la SNCF, dont il a du reste fallu réduire la variété des situations possibles pour que le logiciel fonctionne correctement et que le client puisse s'adapter. De nombreux autres exemples peuvent être cités comme le calcul des salaires (multiplicité des situations concernant le temps de travail, les rémunérations de plus en plus individualisées, les prélèvements comme la CSG déductible et non déductible), le nombre de références que gère

la grande distribution, les différentes formules de prêt des banques, les modalités d'obtention des diplômes…

C - DES CHANGEMENTS TECHNIQUES PERMANENTS

Sur une période relativement courte, l'économie du logiciel se caractérise par la permanence des changements techniques qui tout à la fois constitue une réponse à la complexité croissante des problèmes à résoudre et permet le développement de cette complexité (complexité proactive). Les principaux changements sont tout d'abord présentés (1). Le fait que ces changements concernent l'ensemble des modèles d'innovations que recense l'économie de l'innovation confirment leur richesse et leur diversité (2). Les conséquences d'une telle profusion d'innovations sont une instabilité technologique permanente (3).

1 - Des changements techniques dans tous les aspects de la production des logiciels…

Les multiples changements techniques concernent aussi bien l'extension accélérée des domaines d'applications des logiciels, la création de nouveaux produits, le renouvellement des méthodes de production, les innovations architecturales, les innovations dans la programmation et l'apparition de nouveaux langages, et l'utilisation d'outils d'automatisation (génie logiciel).

a - L'extension des domaines d'application

L'extension des domaines d'application des logiciels s'effectue schématiquement selon trois dimensions (l'automatisation, la communication et la modélisation) et affecte une partie de plus en plus importante de l'activité économique et sociale, avec une complexité croissante des problèmes que doivent résoudre les logiciels.

Tableau VI

Dimension	**Du plus simple au plus complexe**		
Automatisation	Application de règles simples	Système-expert, systèmes d'aide à la décision.	Gestion de la relation client

Communication	Présentation, P.A.O.	Multimédia, site Web	*Groupware*, Commerce électronique
Modélisation	Tableur	C.A.O.	Réalité virtuelle

Concernant l'automatisation, celle-ci a d'abord concerné l'application de règles simples dans les domaines de la gestion et du pilotage de processus[10], puis des tentatives de simuler les comportements et raisonnements effectués par des personnes réelles pour résoudre un problème se rapportant à un domaine d'activité précis (système expert[11]). Ces systèmes remplissent, avec des succès différenciés selon les domaines, des fonctions de diagnostic, de conception, de prévision ou d'optimisation. Plus récemment c'est dans le domaine de la gestion de la relation client que se multiplient les efforts d'automatisation : recueil automatique de multiples informations sur les clients, établissement de relations significatives entre ces critères et propositions personnalisées (en termes de produits, voire en termes de prix différenciés).

Concernant la communication, l'informatisation a permis d'améliorer la présentation d'informations de plus en plus variées (texte, image, plan, schéma…) et l'intégration d'informations situées sur les supports les plus divers (multimédia). Elle vise maintenant à faciliter les relations de coopération entre membres d'une même organisation (*groupware*[12]) et de coordination entre différentes entreprises (places de marché électroniques pour le commerce entre entreprises notamment).

Concernant la modélisation, l'informatique est devenue un outil courant dans la représentation de situations simples (à l'aide d'un tableur par exemple). Elle est utilisée pour la création d'objets virtuels (biens mais aussi services) qui constituent une aide précieuse dans

[10] Récemment la gestion informatique de processus s'est étendue à la circulation automatique des documents (*workflow*).

[11] Certains auteurs préfèrent parler de système à base de connaissances, laissant la qualité d'expert à la personne qui est détentrice d'un ensemble homogène de connaissances (Luc Rubiello, 1997, p. 17).

[12] Le *groupware* est l'ensemble des méthodes, procédures, logiciels et plates-formes informatiques permettant à des personnes, associées dans un même contexte professionnel, de travailler ensemble avec le maximum d'efficacité.

les activités de conception. Un pas supplémentaire est franchi avec l'utilisation d'une "réalité virtuelle" dans des situations problématiques (simulateur de vol, explosion nucléaire…) ou à des fins esthétiques (effets spéciaux dans le cinéma par exemple).

Bien évidemment un même domaine d'application peut connaître une informatisation qui concerne les trois dimensions : par exemple, la gestion informatisée des stocks comporte des aspects d'automatisation (actualisation des stocks en fonction des achats et des ventes), de communication (visualisation de l'état des stocks, mécanismes d'alerte en deçà d'un certain seuil) et de modélisation (optimisation du niveau des stocks en fonction des prévisions). De même, la plupart des logiciels ont des fonctions qui couvrent ces trois dimensions : un banal traitement de texte est un outil de communication, qui dispose de fonctions automatisées (mise en forme, vérification orthographique et grammaticale, résumé, table des matières) et qui permet une certaine modélisation de la pensée de l'auteur (vision instantanée des effets d'une modification, mode "plan", liens hypertextes..).

b - La création de nouveaux produits

L'extension des domaines d'applications suscite la création permanente de nouveaux types de logiciels. Par exemple, la gestion automatisée de la relation clients suppose la constitution de gigantesques entrepôts de données (*data warehouse*) et la création d'outils logiciels (*data mining*) permettant de les exploiter efficacement. Le développement du *groupware* nécessite le développement de logiciels (parfois appelés collecticiels) et l'adjonction de fonctions spécifiques à des logiciels bureautiques permettant à un réseau de personnes bien identifiées d'échanger des informations et de créer collectivement des documents en groupe et en temps réel. La création de places virtuelles d'échanges entre entreprises nécessite la mise en place d'applications partagées entre plusieurs entreprises (*crossware*). Le développement de fonctions plus complexes voit la mise en place d'agents logiciels "intelligents", spécialisés dans un certain nombre de tâches (par exemple la recherche d'informations adéquates) et capables de se perfectionner par apprentissage, leur rôle étant

d'interpréter, de façon toujours plus pertinente, les intentions de l'opérateur et d'exécuter, en large autonomie, des séquences complexes d'opérations[13].

La création de nouveaux domaines d'application nécessite également le développement d'outils qui ont une utilité indirecte. Par exemple le développement des réseaux a suscité l'apparition de logiciels de cryptographie, de compression de données, de sécurité (*firewalls*) et la constitution d'annuaires électroniques universels (comme *Novell Directory System*) qui permettent de gérer automatiquement les profils des utilisateurs (avec les autorisations correspondantes), les adresses électroniques, les chemins d'accès…

Le développement de nouveaux produits peut aussi consister en des améliorations significatives des produits existants. Par exemple, la plupart des Systèmes de Gestion de Bases de Données qui étaient à l'origine "hiérarchiques" se sont transformés en SGBD "relationnels" pour pouvoir gérer la plus grande complexité des relations existant entre les données. L'utilisation des différentes applications a été grandement facilitée par l'introduction d'une interface graphique, appelée WIMP (*Windows, Icons, Menus, Pointer*), que Frederick P. Brooks considère comme "une des innovations logicielles les plus remarquables de ces vingt dernières années" (1996, p. 227).

Enfin, l'apparition de nouveaux produits peut résulter de l'intégration dans un seul produit, de fonctions auparavant effectuées par des produits distincts (progiciels intégrés de gestion d'entreprise, suite bureautique intégrée).

c - Le renouvellement des méthodes de production des logiciels

La question des méthodes de production des logiciels est peut-être le domaine qui a connu les changements les plus rapides suivant une évolution assez semblable à celle de l'ensemble de l'économie sur une longue période[14]. D'une quasi-absence de méthode formalisée caractéristique d'une production de type artisanal, la production de logiciels a vu l'adoption de méthodes de plus en plus formalisées avec un découpage de plus en plus fin en phases séparées que l'on peut rapprocher des principes de l'Organisation Scientifique du

[13] Par exemple pour accomplir des tâches comptables, ont été développés des "robots comptables" qui sont des agents pouvant exécuter automatiquement des procédures logicielles complexes dans la comptabilité.

Travail. La complexité croissante du dessin des lettres qui représentent les différentes phases de ces méthodes (méthode en V, puis en Y et enfin en W) témoigne de l'augmentation du nombre d'étapes qui s'exécutent séquentiellement. Dans ce type de méthodes, les relations entre les développeurs et les utilisateurs (ou ceux qui les représentent) se limitent aux phases les plus en amont (définition des besoins et spécifications du logiciel) et les plus en aval (validation du logiciel). L'extrême importance des phases amont, où toute imperfection entraîne des coûts de correction ou de modification très élevés, a conduit à lui consacrer des méthodes spécifiques : par exemple, la méthode JAD (*Joint Application Design*) élaborée par IBM en 1977, qui consiste à réunir et à faire travailler ensemble les utilisateurs et les informaticiens avec comme objectif des décisions prises en commun et des spécifications acceptées par l'ensemble des participants.

Malgré ces efforts, l'allongement des processus de développement et l'insatisfaction terminale fréquente des utilisateurs[15] a conduit à l'adoption de *méthodes itératives*, permettant des interactions permanentes ente concepteurs et utilisateurs et dont l'objectif est une adaptation plus rapide et plus précise aux besoins. Un premier pas dans cette direction a été la modification du "modèle de la cascade" - dénommé ainsi parce qu'à chaque étape il était seulement possible de descendre à l'étape suivante - par Winton Royce (1970) en incluant une rétroaction d'une étape vers la précédente et une limitation de cette rétroaction à l'étape qui précède immédiatement, de façon à limiter les coûts et les délais qu'elle entraîne (Frederick P. Brooks, 1996, p. 230). Un modèle qui approfondit cet objectif est le modèle du développement "en spirale" de Barry W. Boehm, qui introduit la notion de cycle de développement itératif, avec des possibilités de retour et de modification à tous les niveaux et pas seulement à la phase immédiatement précédente. L'aboutissement de cette évolution est constitué par les méthodes évolutives où les phases de spécification (quoi faire ?) et de conception (comment faire ?) sont étroitement imbriquées. Ces méthodes, moins formalisées, reposent sur le prototypage rapide (un prototype est utilisé pour clarifier les besoins des

---

[14] La *Software Productivity Research Inc* identifie 65 méthodes différentes de développement de logiciel (Warnier-Orr, Yourdon, Jackson, Merise, RAD…).

[15] "Ce n'est qu'après avoir achevé la construction du système qu'on peut découvrir qu'il est trop malcommode à utiliser, qu'il a des performances inacceptables, ou qu'il est dangereusement exposé aux erreurs ou aux stratagèmes des utilisateurs (..). L'examen des spécifications est censé découvrir ce genre de problème au début du processus, mais le test d'utilisation réel est irremplaçable" (Frederick P. Brooks, 1996, p. 231).

utilisateurs en leur permettant de réagir au vu du système[16], et pour définir le système final), et le prototypage évolutif (ou développement incrémental) où, à partir d'une première version rapidement développée et aux fonctionnalités limitées, sont créées des versions successives utilisables, par l'intégration d'éléments nouveaux (complexification), la suppression d'éléments inutiles (simplification), la prise en compte de nouveaux aspects du problème (divergence), et l'intégration d'éléments du système restés déconnectés (convergence). Outre les possibilités de rectification en cours de développement et la motivation que constitue la vision permanente de la concrétisation des efforts accomplis, l'intérêt de ces méthodes est qu'elles permettent une répartition du pouvoir plus équilibré entre les concepteurs et les utilisateurs, qui "forment une équipe de résolution de problèmes" (Robert Michon, Lin Gingras, 1988, p. 129) et qu'elles facilitent l'intercompréhension mutuelle, grâce à l'existence de versions précoces qui créent "un pont sémantique entre concepteur et utilisateur" (idem, p. 130).

d - Les innovations dans la programmation et l'apparition de nouveaux langages

La programmation, malgré sa jeunesse, a connu des innovations importantes qui se sont le plus souvent traduites par l'apparition de nouveaux langages de programmation, un langage de programmation n'étant rien d'autre qu'un ensemble déterminé et fixe d'instructions qui permettent la transcription du problème considéré.

La première grande innovation a été l'introduction de la *programmation structurée*, au début des années soixante-dix, avec notamment le langage Pascal, qui visait à éviter les instructions de branchement (le célèbre Go To dont la nocivité avait été mise en évidence dans un article célèbre d'Edsger Djikstra en novembre 1968) grâce à des structures conditionnelles (si… alors…sinon) et itératives (les boucles pour, jusqu'à, tant que). L'objectif est d'avoir un schéma de raisonnement prédéfini et des entrées et sorties prédéfinies, et de conduire les opérations de façon rationnelle et prédéterminée, par opposition aux méthodes

[16] Les documents produits lors des premières phases de nombreuses méthodes, par exemple les modèles conceptuels de données de la méthode Merise, sont souvent trop complexes et trop abstraits pour l'utilisateur (Robert Michon, Lin Gingras, 1988, p. 127). Il est difficile d'exprimer des besoins précis sans voir le système, comme il est difficile de spécifier sans apprécier les problèmes de réalisation (Jean-Marc Geib, 1989, p. 17).

d'improvisation. En forçant l'exposition de la structure du raisonnement suivie jusque dans le code source du programme, la programmation structurée permet de contrôler la rigueur des raisonnements et doit rendre plus facile la compréhension et la modification des programmes.

La deuxième innovation qui prolonge la précédente a été l'introduction de la *programmation modulaire* en découpant un programme en une arborescence hiérarchisée de multiples modules et sous modules. C'est une conception fonctionnelle descendante, dans laquelle on s'efforce de minimiser les communications et les contrôles entre modules, et de maximiser la cohésion interne de chaque module individuel. On peut y voir une illustration de la théorie des systèmes complexes de Herbert A. Simon (1974), où celui-ci montre l'efficacité résultant d'une décomposition hiérarchique et de l'utilisation de formes intermédiaires stables.

La troisième innovation est la naissance de la *programmation orientée objet*s, apparue avec le langage Smaltalk, qui est basée sur les notions de *classe*, d'*héritage* et d'*encapsulation*. A l'opposé de la programmation classique qui sépare données et procédures, l'approche orientée-objet supprime cette dichotomie en intégrant données et fonctions de traitement (appelées méthodes) dans des entités appelées classes. Une *classe* est définie par une structure, des propriétés et un ensemble de méthodes utilisées pour manipuler les objets qui sont les instances de cette classe. L'*encapsulation* est le fait de masquer la structure interne et les traitements d'un objet (partie privée), l'accès à cet objet ne pouvant être effectué que par l'intermédiaire de messages activant les méthodes définies dans son interface publique. L'*héritage* est le fait qu'une classe peut être définie par extension ou spécialisation d'une autre classe (héritage simple) ou de plusieurs autres (héritage multiple). L'intérêt principal de la programmation objet est qu'elle permet une meilleure maîtrise de la complexité des systèmes, qu'elle facilite la réutilisation des composants logiciels déjà développés et qu'elle permet le développement rapide d'applications, notamment en utilisant les méthodes évolutives précédemment exposées.

Enfin, on peut mentionner plus récemment la conception de "logiciel modulaire" (OCDE, 1997 B, p. 185) qui vise à reproduire les avantages de la programmation orientée objet en les rendant accessibles aux utilisateurs comme aux programmeurs. L'idée est de développer des éléments modulaires comme les *applets* Java de Sun, les composants ActiveX

Face à ces difficultés, "le but d'un prototype est de rendre tangible la structure conceptuelle spécifiée, de telle manière que le client puisse en tester la cohérence et l'ergonomie" (Frederick P. Brooks, 1996, p. 172).

de Microsoft ou OpenDoc d'Apple, qui sont téléchargés lorsqu'ils sont utilisés (donc mis à jour automatiquement), qui peuvent être associés et assortis sur n'importe quel type de matériel parce qu'ils partagent un ensemble d'interfaces standard. L'utilisation de ces éléments de petite taille, qui remplissent chacun une fonction limitée mais qui interagissent étroitement avec d'autres éléments, doit permettre d'adapter le logiciel aux besoins précis de l'utilisateur à un moment donné. La généralisation de ce nouveau modèle à de nombreux domaines dépendra de façon décisive de la réussite du processus de standardisation des interfaces de ces composants.

e - L'utilisation d'outils d'automatisation : le génie logiciel

La production de logiciels a été analysée comme une succession de processus de codifications. L'allongement de ce processus par la création d'étapes intermédiaires lui confère une plus grande efficacité (cf. chapitre I), les étapes les plus en aval pouvant être réalisées automatiquement (par exemple un compilateur qui traduit automatiquement un programme écrit en langage de haut niveau[17] en langage machine). L'idéal, impossible à atteindre, serait de pouvoir programmer directement en langage naturel, des "outils" réalisant automatiquement la conversion en langage machine. De ce point de vue, on peut dire que "la programmation automatique a toujours été un euphémisme désignant la programmation à l'aide d'un langage de plus haut niveau que ce dont dispose le programmeur" (Frederick P. Brooks, 1996, p. 167). L'efficacité de l'allongement du détour de production est un phénomène habituel dans la production économique, mais qui est amplifié par une particularité des outils logiciels : à la différence d'un outil matériel comme une machine, sa reproduction a un coût négligeable, une fois qu'il a été mis au point.

De ce fait, les efforts pour développer des outils logiciels de plus en plus sophistiqués sont constants depuis les débuts de la programmation. Ils se sont amplifiés avec la naissance du génie logiciel, "domaine controversé s'il en est, qui après avoir fait beaucoup rêver dans les années 1980, a aussi beaucoup déçu, avec des gains de productivité qui n'étaient pas au rendez-vous, et des échecs industriels patents" (Jacques Printz, 1998, p. 232). Le génie logiciel, terme apparu en 1968 à l'initiative du Département de la Défense des Etats-Unis, est

[17] Un langage de plus haut niveau est un langage qui a un vocabulaire plus étendu, une syntaxe plus compliquée, et une sémantique plus riche. Le langage de plus bas niveau est le langage machine, le langage de

un ensemble de méthodes et de techniques qui visent à convertir l'activité de réalisation de logiciels, d'un artisanat mystérieux et faillible en une discipline d'ingénierie (OCDE, 1991, A, p. 13). Il comprend l'ensemble des activités de conception et de mise en œuvre des produits et des procédures tendant à rationaliser la production du logiciel et son suivi. Un de ses objectifs est d'accentuer l'automatisation de la production des logiciels, "l'ambition avouée [étant] de faire disparaître les programmeurs au profit des générateurs de codes, ce que proposent déjà certains ateliers de génie logiciel" (Patrick Jaulent, 1992, p. 29)[18].

On peut distinguer dans le génie logiciel deux dimensions. La première est une démarche mathématique et scientifique, très structurée, qui à partir de l'utilisation de méthodes formelles vise à éliminer les "mauvaises pratiques" génératrices d'erreurs et d'ambiguïtés. Un exemple en est la preuve formelle de programmes pour démontrer mathématiquement qu'un programme est correct. La deuxième dimension, plus pragmatique, consiste en en une activité pratique de développement d'outils de programmation visant à accroître la productivité des réalisateurs de logiciels (Génie Logiciel Assisté par Ordinateur).

Sont ainsi apparus au début des années quatre-vingt, une grande variété d'outils logiciels ou outils CASE (*Computer-Aided Software Engineering*) qui couvrent l'ensemble des phases de la production des logiciels : analyse des besoins, conception, estimation du coût et du travail, gestion de projet et documentation, production automatique de programmes et essai, vérification et validation des systèmes. Au sein de ces outils, on distingue les outils *Lowe Case*, centrés sur les activités de programmation et de test, des outils *Upper Case* qui correspondent aux phases amont d'analyse et de conception.

Au-delà de l'utilisation d'outils isolés, une tentative plus ambitieuse consiste à fournir un ensemble intégré d'outils et de méthodes, ou I-CASE (*integrated CASE*), suite d'outils CASE homogènes et cohérents qui permettent de traiter tous les aspects d'un projet (phases du cycle de vie, gestion de projet, analyse), et dont le plus connu est AD-Cycle d'IBM.

---

plus haut niveau est le langage naturel, les différents langages de programmation occupant des positions intermédiaires entre ces deux extrêmes.

[18] La recherche d'une automatisation croissante de la production est, dans le cas des logiciels, particulièrement recherchée en raison des qualifications élevées des informaticiens, et de leur relative pénurie face à l'importance de la demande, qui expliquent des rémunérations élevées.

2 - …qui correspondent à l'ensemble des modèles d'innovations de certaines théories évolutionnistes.

A partir de l'étude de l'innovation dans les services, Faïz Gallouj et Olivier Weinstein (1997) ont élaboré une typologie des innovations, dont un des intérêts est qu'elle permet de mettre en évidence la richesse des innovations dans les services, que des approches plus "technologistes" ne perçoivent pas. Cette typologie est construite à partir d'une représentation d'un service comme un système de caractéristiques et de compétences, représentation que nous avons mobilisée pour analyser les caractéristiques des logiciels (cf. chapitre I). *Or quand on examine les innovations dans le domaine des logiciels, on constate qu'elles concernent tous les types d'innovations mis en évidence par Faïz Gallouj et Olivier Weinstein, ce qui témoigne de la forte dynamique innovatrice de l'économie du logiciel*.

Elle est illustrée tout d'abord par la fréquence des *innovations radicales*, qui concernent la création d'un produit totalement nouveau (par exemple le tableur ou plus récemment le navigateur), la création d'une architecture fondamentalement différente (par exemple le client-serveur) ou d'une nouvelle façon de programmer (par exemple la programmation orientée objet). Mais les innovations radicales ne constituent qu'une petite partie des innovations, les autres innovations (innovations incrémentales) étant différenciées en plusieurs catégories dans la typologie de Faïz Gallouj et Olivier Weinstein.

La première catégorie concerne les *innovations d'amélioration* qui consiste simplement à améliorer certaines caractéristiques. Dans le cas des logiciels, ce type d'innovations est fréquent et important : par exemple, la possibilité de gérer un beaucoup plus grand nombre d'enregistrements dans le cas d'un système de gestion de bases de données, ou la forte augmentation du nombre de couleurs et de niveaux de détail que peut traiter un logiciel de retouche d'images ou de photographies.

La deuxième catégorie concerne les *innovations "incrémentielles"* qui s'opèrent par transformation ou adjonction de caractéristiques. Pour les logiciels, un exemple spectaculaire de transformation d'une caractéristique importante (la facilité d'utilisation) est l'introduction d'une interface graphique. L'adjonction de caractéristiques est réalisée par l'introduction de nouvelles fonctionnalités à chaque version (fréquente) d'un logiciel, que facilite la nature particulière d'un logiciel.

La troisième catégorie, appelée *innovation de recombinaison*, consiste à jouer sur les possibilités de combinaisons multiples de différentes caractéristiques. Une illustration est la création d'un produit logiciel unique en intégrant différents produits existants (progiciel intégré bureautique ou de gestion d'entreprise). Plus récemment, la conception de logiciels modulaires, exposée précédemment, est significative des possibilités et des potentialités de ce type d'innovations pour les logiciels.

La quatrième catégorie est représentée par des *innovations ad hoc*, qui est définie comme la construction sociale interactive d'une solution au problème particulier d'un client donné, souvent coproduite et reconnue a posteriori, non reproductible en tant que telle puisqu'elle est localisée et contextualisée, mais qui peut servir pour d'autres prestations. Ce type d'innovations concerne plus le développement de logiciels sur mesure, où des méthodes et des schémas de résolutions de problèmes originaux peuvent être réutilisés. Avec le développement de la programmation orientée objet, ce type d'innovations peut s'étendre à des composants logiciels développés.

3 - Quelques conséquences de cette profusion d'innovations : une instabilité technologique permanente

Il faut distinguer le rythme soutenu d'apparition d'innovations, de leur processus de diffusion qui peut être lent pour plusieurs raisons. Tout d'abord, il faut tenir compte du temps nécessaire pour assimiler une innovation et effectuer l'apprentissage de son utilisation[19]. De plus, l'apprentissage a un caractère ambivalent : plus il est performant dans une direction, plus il est freiné dans les autres directions (Olivier Favereau, 1998, p. 199). De ce point de vue, une volonté de changements trop rapides peut être contre-productive. Serge Bouchy souligne que "l'intégration perpétuelle de nouvelles techniques sur lesquelles l'état des connaissances n'en est parfois qu'aux balbutiements, (…) freine l'assimilation et le réel transfert des connaissances" (1994, p. 143). De même Jacques Printz estime que la frénésie innovatrice à

[19] Capers Jones cite une étude non publiée d'IBM qui montre que "le transfert de technologies est un processus terriblement lent". Cette étude concluait que la diffusion d'une technologie est une fonction du temps analogue à une suite de Fibonacci : une nouvelle méthode met un an pour toucher 15 %, trois ans pour 50 % et cinq ans pour monter à 90 % (1989, p. 276). Ce processus peut être beaucoup plus long pour une innovation majeure : il s'est écoulé vingt ans entre les premiers langages orientés objets dans les laboratoires et l'apparition de produits commerciaux effectivement utilisés.

laquelle on assiste depuis le début des années quatre-vingt-dix est un véritable obstacle à une croissance en quantité et en qualité des logiciels" (1998, p. 323)[20].

L'importance des effets d'apprentissage peut expliquer pourquoi des innovations porteuses de solutions à long terme ne sont pas adaptées en raison de leurs coûts immédiats importants. Ces coûts comprennent le coût de l'innovation elle-même[21], et les coûts de migration et de formation, qui peuvent être élevés quand l'innovation représente une discontinuité dans le développement technologique : par exemple, l'adoption du génie logiciel représente un changement de paradigme dans les méthodes de réalisation du logiciel avec des techniques, des compétences, des méthodes, des outils et des modes de travail radicalement nouveaux (OCDE, 1991 A, p. 30)[22].

Dans certains cas, des rigidités sociales et institutionnelles peuvent également entraver l'introduction d'innovations. Il faut également prendre en compte, notamment quand il existe des incompatibilités entre technologies, la crainte d'effectuer un mauvais choix, d'introduire une innovation qui sera rapidement dépassée par une autre. Paradoxalement, "lorsqu'une évolution rapide est escomptée, cette expectative elle-même ralentit l'application des nouvelles techniques ; dans un monde en évolution rapide, les gens tendent à adopter une politique attentiste" (Arnold Heertje, 1988, p. 11). Or dans l'économie du logiciel, et plus généralement dans l'informatique, il est difficile de faire des pronostics sur les évolutions même à un horizon de court/moyen terme.

Il en résulte une très grande hétérogénéité technologique avec des processus de diffusions qui peuvent avoir des rythmes très différents selon les acteurs, et une économie du logiciel qui semble condamnée à une instabilité technologique permanente : "la rapidité de

[20] F.G. Roux ironise sur "les objets autres nouveautés miraculeuses comme les langages de quatrième génération, les ateliers de génie logiciel, l'intelligence artificielle, les systèmes experts ; la réponse unanime des responsables informatiques est que « l'avenir de l'informatique passe par l'orientation objet qui est la seule solution qui sera porteuse de productivité et de qualité » ; il y a 10 ans, ils disaient que ce serait Merise, 8 ans les AGL et l'IA, 5 ans Unix, 3 ans le modèle client-serveur" (L'informatique professionnelle n° 135, Juin Juillet 1995, p.30-31).

[21] Par exemple les très hauts niveaux d'investissement nécessaires expliquent pourquoi les petites organisations productrices de logiciels n'utilisent pas les ateliers de génie logiciel, y compris certaines sociétés qui ont participé à la mise au point de ces AGL (OCDE, 1991 A, p. 26).

[22] Emmanuel Saint-James souligne que l'habitude de la communauté informatique d'introduire chaque nouvelle idée par un langage nouveau, oblige "à se mettre dans la tête une multitude d'idiosyncrasies arbitraires et à réécrire les bibliothèques de programmes" (1993, p. 4).

développement de technologies nouvelles empêche constamment toute velléité de stabiliser les choix technologiques. Le cycle expérimentation puis rationalisation et stabilisation est ainsi constamment réactivé par l'offre dont les effets d'annonce fonctionnent d'autant plus comme des injonctions d'achat qu'une forte incertitude règne sur l'environnement technologique futur" (Alain Rallet, 1997, p. 86).

Une illustration est la profusion des langages informatiques utilisés : par exemple, alors que de multiples tentatives ont existé pour promouvoir un langage universel soutenu par des acteurs puissants (PL/1 par IBM à l'époque où elle dominait l'informatique mondiale, puis ADA par le Département de la Défense des Etats-Unis avec un investissement qui a dépassé le milliard de dollars, JAVA actuellement par Sun), la diversité des langages utilisés n'a cessé de s'accroître, et un langage comme Cobol reste très utilisé alors que "depuis le début des années soixante-dix, au moins, on annonce pour prochaine sa fin" (Jean Bres, 1994, p. 5)[23].

Nous avons vu que le logiciel était un objet particulièrement difficile à appréhender sur un plan technique. Son analyse sur un plan économique est au moins aussi délicate, en raison des difficultés pour lui appliquer les catégories économiques habituelles.

Section II - L' « insaisissabilité » théorique : les difficultés à appliquer les catégories économiques habituelles

En tant que produit, le logiciel est difficilement appréhendable par les catégories économiques habituelles : il occupe une position intermédiaire entre les biens et les services (A), il possède certains attributs des biens collectifs (B), il est difficile de lui appliquer les théories de la valeur qui ont été forgées par les économistes pour expliquer l'évolution des prix (C).

[23] Cet auteur cite une étude de Micro-focus qui estime qu'il existe dans le monde 100 milliards de lignes de code, et que sur ces 100 milliards, 77 % sont écrites en Cobol.

A - UN PRODUIT HETEROGENE OCCUPANT DES POSITIONS DIVERSES SUR LE CONTINUUM BIENS-SERVICES

L'hétérogénéité traitée dans ce paragraphe concerne la variété des formes technico-économiques des logiciels et non la diversité des fonctions des logiciels analysée précédemment. Sous cet aspect, les logiciels peuvent se positionner à différents endroits du continuum existant entre les biens et les services. Après avoir analysé les tendances à une fusion entre les biens et les services (1), nous verrons la possibilité et la nécessité de maintenir une distinction entre les biens et les services, en intégrant une différenciation entre les biens tangibles et les biens intangibles (2), que nous utiliserons pour caractériser les différentes formes de logiciels (3).

1 - Vers une fusion des biens et des services ?

Certaines analyses ont mis en évidence une quasi-disparition des frontières entre les biens et les services, à partir du constat de la régression des services purs (nécessitant peu d'appuis matériels) et des biens purs (n'intégrant pas de services). Ces analyses s'appuient sur l'existence de deux tendances : la tendance à l'industrialisation des services avec le développement de processus "d'objectivation" et de standardisation des services, liés notamment à leur informatisation, dont une forme extrême est la production de quasi-biens (Faïz Gallouj, 1997, p. 30) ; la tendance à la "servicialisation" des biens comportant une part croissante de services et intégrant une relation de service dans une volonté de répondre plus finement aux besoins du client (production sur mesure). Certains auteurs infèrent de ces tendances l'absence de distinction fondamentale entre les biens et les services : par exemple, André Barcet, Joël Bonamy, et Anne Mayère (1984, p. 128) considèrent la notion de "produit-service" comme une unique catégorie économique pouvant revêtir différentes formes ; C. Belleflamme, J. Houard, B. Michaux et O. Ruyssen (1986) rejettent eux aussi l'opposition bien/service en considérant que ce qui intéresse le client c'est la satisfaction d'un besoin (d'une fonction) ; pour ces auteurs, chaque produit est le résultat, dans des proportions variables, de la combinaison d'un processus de production et d'un processus de servuction (différenciation et adaptation du produit aux spécifications de l'utilisateur).

Sans nier l'intérêt de ces analyses et la réalité des tendances évoquées, il nous semble néanmoins possible et nécessaire de maintenir une distinction entre les biens et les services[24]. Notre position peut se résumer ainsi : à certains égards, cette distinction n'est pas pertinente, mais pour analyser certaines questions, elle le demeure.

2 - Une distinction maintenue entre des biens tangibles, des biens intangibles et des services

Nous reprenons ici l'analyse de Jean Gadrey (1999 A). Tout en rejetant les oppositions classiques et contestables entre les biens et les services fondées sur les critères de matérialité, de stockabilité, de périssabilité, de transportabilité, Jean Gadrey réexamine dans un premier temps la différence de nature existant entre les biens et les services, énoncée par Peter Hill (1997) : seuls les biens sont des entités existant indépendamment de leurs producteurs et de leurs consommateurs ; sur les biens peuvent être établis des droits de propriété ; à l'inverse les services ne sont pas des entités, ce sont des changements de condition ou d'état d'une entité possédée par le demandeur du service, ce qui a comme conséquence que le produit d'une activité de service ne peut être revendu indépendamment de l'entité sur laquelle il porte. A partir de cette opposition, Jean Gadrey effectue deux ajouts importants. Premièrement, la notion d'entité, qui joue un rôle central dans la distinction opérée par Peter Hill entre les biens et les services, mais qui n'est pas vraiment définie, peut être appréhendée à partir de l'existence d'une identité sociale et historique des biens, attestée par la présence de "marques identitaires", en précisant que le processus d'attribution d'une identité à un bien n'est pas simplement technique mais de nature conventionnelle. Deuxièmement, la définition des services comme changement de condition ou transformation d'état d'une entité ne permet pas d'intégrer une partie importante de ce qui est considéré usuellement comme des services (restauration, hôtellerie, spectacles, une partie de l'activité commerciale…). Pour englober l'ensemble du champ des services, Jean Gadrey propose de centrer la définition des services sur la mise à disposition temporaire d'une capacité technique et humaine d'une organisation prestataire à un client-usager, pour produire des effets utiles sur lui-même ou sur des biens qu'il possède.

[24] "En dépit de ces formes de convergence entre les deux modèles, avec un secteur industriel devenant de plus en plus intensif en service à la fois sur le plan de ses process et sur celui de ses produits, il semble prématuré de conclure à une similarité des situations ; (…) la dimension de service des produits industriels reste dans la majorité des cas de type plus périphérique que centrale, et la tendance à l'individualisation est loin d'être répandue" (Jean Gadrey, 1996 A, p. 193).

Dans le même article, Peter Hill introduit une distinction importante au sein des biens, entre les biens tangibles et les biens intangibles. Les biens intangibles sont définis comme étant les "originaux" produits d'une activité de création scientifique, technique ou artistique. Ces "originaux" inscrits sur des supports divers peuvent être facilement dupliqués à un coût de plus en plus faible et généralement sans commune mesure avec le coût de création de l'original. Les biens intangibles se limitent aux cas où "l'original et les copies ont les mêmes caractéristiques utiles" (Jean Gadrey, 1999 A, p. 16). Dans certaines activités de création artistique (peinture, sculpture…) où l'original a une valeur bien supérieure à la copie, les originaux peuvent être considérés comme des biens tangibles non reproductibles. Les biens intangibles sont des biens dans la mesure où ce sont des entités sur laquelle des droits de propriété peuvent être établis, même si ces droits de propriété prennent fréquemment des formes spécifiques (copyright, droit d'auteur) en raison de certaines caractéristiques des biens collectifs que possèdent les biens intangibles. Ces biens sont intangibles en raison de leur immatérialité, de leur absence de dimensions et de coordonnées physiques, ou plus précisément de l'absence d'importance sur un plan économique de leur localisation géographique. Les biens intangibles occupent une place de plus en plus importante dans l'activité économique au niveau de la production (descriptifs d'un produit, d'un composant, d'une méthode de production) et de la consommation finale (texte, film, composition musicale…).

3 - L'application à l'informatique et la situation particulière des logiciels

Ces distinctions sont particulièrement fécondes pour analyser les différentes formes technico-économiques que peuvent prendre les programmes informatiques. Dans des cas extrêmes, le code-objet d'un programme informatique peut être gravé dans le silicium et être intégré de façon indissociable au composant électronique sur lequel il agit. Ces composants électroniques programmés sont des biens tangibles. Il s'agit certes d'une activité où le coût de conception du premier exemplaire est particulièrement élevé mais où la production d'un exemplaire supplémentaire n'est pas une opération triviale. Mais, et c'est la situation la plus fréquente, le code-objet d'un programme informatique peut exister également indépendamment des composants électroniques sur lesquels il agit, être fixé sur un support matériel distinct (cartes perforées, disques magnétiques ou opto-électroniques…), ce qui a

pour conséquence qu'il devra être chargé dans une mémoire (mémoire vive[25]) pour pouvoir être exécuté par les composants électroniques pour lesquels il a été conçu.

Logiciel et progiciel

Il importe dans cette situation de distinguer deux possibilités très différentes. Une première possibilité est le développement par une entreprise (ou par le service informatique interne de l'organisation) d'un logiciel sur mesure en réponse à un besoin précis. Dans ce cas, la production de logiciel peut être considérée comme une activité de services, le prestataire mettant ses compétences à la disposition d'un client-usager pour développer un programme informatique. Les spécificités de ce produit unique et original sont négociées entre le prestataire et le client, l'adéquation aux besoins dépendant étroitement de la qualité de la relation de service qui s'instaure entre eux. La deuxième possibilité est la production préalable d'un logiciel standard, dont les producteurs escomptent qu'il répondra aux besoins d'utilisateurs anonymes, et qui est vendu "sur catalogue". Il s'agit ici typiquement de la production d'un bien intangible, la production d'un nouvel exemplaire strictement identique à l'original étant une simple opération de copie à un coût dérisoire. Cette forme de logiciel est appelée progiciel (par contraction de produit-logiciel), défini comme étant un "ensemble complet et documenté de programmes, conçu pour être fourni à plusieurs utilisateurs en vue d'une même fonction" (Journal Officiel du 7 décembre 1980). Une définition plus développée indique qu'il s'agit "d'un ensemble cohérent et indépendant, comprenant des programmes, des services, des supports de manipulation de l'information conçus pour réaliser des traitements informatiques standard, proposé suivant une forme commerciale et qu'un usager peut utiliser de façon autonome après une mise en place et une formation limitée" (01 Informatique, 24 mai 1983). On peut remarquer que la production de logiciels sur mesure est constitutive d'une logique de demande (qui doit s'exprimer avant le démarrage de l'activité), alors que la production de progiciel repose sur une logique d'offre (mise sur le marché d'une production déjà réalisée) (Christine Babelon, 1987, p. 88).

[25] En informatique, on distingue fondamentalement deux types de mémoires, même si se développent de plus en plus des mémoires "intermédiaires" entre ces deux catégories : les mémoires mortes ou ROM (Read Only Memory) dont le contenu est figé définitivement lors de la production du composant, et les mémoires vives ou RAM (Random Access Memory) dans lesquelles on peut stocker de façon temporaire des programmes, des données... (le contenu est perdu lors de la coupure de l'alimentation électrique).

Le tableau suivant fait apparaître le positionnement différencié des logiciels selon leur nature technico-économique.

Tableau VII : Les logiciels entre les biens et les services.

<table>
<tr><th colspan="2">Biens tangibles</th><th colspan="2">Biens intangibles</th><th colspan="3">Services</th></tr>
<tr><td>Autres biens</td><td>Composants électroniques programmés</td><td colspan="2" rowspan="2">Progiciels

Outils et systèmes Applicatifs</td><td>Logiciels sur mesure</td><td>Services informatisés</td><td>Autres services</td></tr>
<tr><td></td><td></td><td>Autres services informatiques</td><td></td><td></td></tr>
<tr><td></td><td></td><td colspan="2">Produits culturels multimédias</td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td colspan="2">Autres biens intangibles</td><td></td><td></td><td></td></tr>
</table>

Cette représentation appelle deux remarques complémentaires. Premièrement, au-delà des relations de complémentarité entre les composants matériels et logiciels, il existe des possibilités de substitution entre les différentes formes que peut prendre un même programme informatique. Ce constat illustre la remarque plus générale de Michel Callon (1993, p. 25-26) selon lequel "il n'existe en réalité aucune barrière infranchissable entre un énoncé, une machine ou un corps discipliné, les substitutions et les réinscriptions [étant] toujours possibles, au moins dans une certaine mesure". Un exemple historique célèbre est le calcul des logarithmes, que Charles Babbage voulait faciliter en le décomposant en différentes opérations dont les plus simples pourraient être réalisées automatiquement par une machine qu'il était en train de concevoir. Cette tâche a été effectuée par des logiciels dès l'apparition des premiers ordinateurs, elle est maintenant intégrée dans des composants électroniques présents sur la moindre calculatrice de poche. Pour prendre un exemple actuel, si l'on considère le secteur des jeux informatiques, secteur particulièrement typique de cette

diversité, on constate qu'un même jeu informatique peut être réalisé de multiples façons. Ce jeu peut reposer uniquement sur des composants électroniques (machine dédiée), comporter une partie matérielle (la console de jeu et son système d'exploitation) et une partie progicielle (le jeu lui-même), fonctionner sur un micro-ordinateur où le matériel se caractérise par son universalité, le système d'exploitation et le jeu étant réalisés sous forme de progiciels. Mais ce même jeu peut également avoir été écrit spécifiquement par l'auteur-utilisateur sur un micro-ordinateur comportant un système d'exploitation standard, voire dans le cas des premiers micro-ordinateurs vendus en kit sans aucun programme existant, avoir nécessité le développement par l'utilisateur d'un système d'exploitation puis du jeu lui-même.

Deuxièmement, les frontières entre les différentes catégories ne sont pas étanches et le rattachement d'une production concrète à une activité n'est pas toujours évident, ce qui pose des problèmes d'interprétation des statistiques existantes.

Entre le logiciel sur mesure et le progiciel, il existe des produits intermédiaires dérivés des produits standards : progiciels paramétrables pour être adaptés aux spécificités de l'entreprise cliente, modules vendus séparément et fournis dans le cadre d'une prestation intégrant des services. La STERIA, qui la première en France a développé ce type d'activités, parle de semi-produits pour désigner ces solutions constituées d'un noyau commun et d'une partie sur mesure, dont elle estimait qu'elle revenait deux fois moins chère qu'une solution intégralement sur mesure (Eric Labat, 1984, p. 248). De ce fait, il est parfois difficile de délimiter le champ statistique des progiciels de celui des logiciels sur mesure, d'autant que face aux reproches adressés aux SSII françaises de ne pas faire de progiciels, certaines ont tendance à rebaptiser à des fins commerciales "progiciels" la simple existence de "quelques briques de base", l'essentiel étant pourtant écrit sur mesure (Gérard Dréan, 1996 A, p. 278).

Les services informatiques

Parmi les services liés à l'informatique, il est nécessaire d'opérer une distinction entre les services informatisés, qui avaient une existence antérieure à leur informatisation, même si celle-ci a pu transformer plus ou moins profondément la nature de l'activité, par exemple la comptabilité, et les services informatiques qui sont des services de support à l'activité informatique.

Les services de support comprennent d'une part des services basés sur des engagements contractuels de moyens (conseil, assistance, formation) et d'autre part des services reposant

théoriquement sur des engagements de résultats (développement de logiciel, ingénierie des systèmes et intégration de systèmes). L'ingénierie de systèmes se distingue du développement de logiciels par la fourniture d'un système complet comprenant du matériel. Elle est appelée intégration de systèmes si le matériel comprend des éléments hétérogènes en provenance de constructeurs différents. Les services informatiques constituent un "monde varié et instable, parallèle à celui des produits, auquel il est lié par des relations complexes de complémentarité et de concurrence" (Gérard Dréan, 1996 A, p. 260). L'évolution est rapide avec l'apparition et la disparition périodiques de certaines catégories de services. Par exemple, la tierce maintenance applicative (confier à un prestataire la montée en charge et l'évolution d'une nouvelle application), apparue récemment, se développe rapidement. En sens inverse, les prestations matérielles ou prestations-machine (traitement à façon, télétraitement par lots à distance, télétraitement en temps réel, vente d'heures machines) qui représentaient l'essentiel des services informatiques au début de l'informatique ont quasiment disparu au profit de prestations intellectuelles (Christine Babelon, 1987, p. 68-69)[26]. Ces prestations matérielles constituaient le "service bureau" où les travaux des clients étaient traités sur les matériels du prestataire. Le service-bureau résultait de l'équipement en ordinateurs du service bureau classique qui existait depuis plusieurs décennies. Avec l'équipement progressif en matériel informatique des entreprises, le service-bureau a cessé d'exister entraînant la disparition ou la reconversion des premières sociétés de service qui s'étaient créées à cette occasion. Il faut toutefois noter que le récent développement du *"facilities management"*, consistant pour une entreprise à faire sous-traiter son informatique à une société de service, redonne une certaine jeunesse peut-être temporaire à ce type d'activités (cf. chapitre VI). Dans l'ensemble des services informatiques, le développement de logiciels occupe une place importante, mais qui n'est pas toujours évidente à mesurer séparément.

Des problèmes de délimitation

Parmi les progiciels, on peut distinguer les progiciels outils et systèmes, des progiciels applicatifs. Ce sont les logiciels outils et systèmes qui, de par leur plus grande proximité avec

[26] Le traitement de données qui représente environ 15 % des services informatiques, en baisse régulière au rythme de 1 à 2 % par an, est actuellement essentiellement constitué de services d'application informatisés comme la gestion des transactions par cartes de paiement, ou les services aux experts comptables ou aux agents d'assurances. Ces prestations figurent dans les services informatiques lorsqu'ils sont fournis par des sociétés considérées comme des sociétés de services informatiques, alors que des prestations comparables n'y figurent pas lorsqu'elles sont fournies par d'autres entreprises. Selon Gérard Dréan, "la présence du traitement dans les services en informatique est essentiellement un archaïsme statistique" (1996 A, p. 268).

le matériel et ses producteurs, ont été le plus rapidement et le plus massivement fournis sous forme de progiciels (phénomène de standardisation), les logiciels d'applications n'étant concernés que plus récemment (notamment avec l'explosion de la microinformatique) et restant plus proches des producteurs de service. La séparation entre composants matériels et composants logiciels n'est pas toujours évidente, notamment avec l'apparition de la microprogrammation qui a introduit un niveau intermédiaire (le microcode) entre les fonctions directement réalisées par les circuits électroniques et celles réalisées par programmation : un microprocesseur contient actuellement une quantité importante de logiciels. De ce fait, la distinction traditionnelle entre matériel et logiciel devient de plus en plus conventionnelle (Gérard Dréan, 1996 A, p. 21). Enfin, l'importance et la pertinence de ces délimitations sont perçues différemment selon le type d'acteurs concerné. Pour un utilisateur final non informaticien, ce qui importe ce sont les services fournis globalement par le système informatique et il lui est indifférent de savoir ce qui dans ces services dépend de dispositifs matériels ou logiciels. Pour un développeur d'applications la question importante est de connaître les fonctionnalités offertes globalement par la "couche" constituée du matériel et de son système d'exploitation, la répartition entre ce qui relève du matériel ou du logiciel de base n'ayant qu'une importance secondaire pour l'optimisation éventuelle des performances de l'application. Par contre, cette séparation reste importante et fait l'objet d'arbitrages technico-économiques délicats pour ceux qui conçoivent l'architecture du matériel d'une part, l'écriture du système d'exploitation d'autre part. Enfin, pour l'économiste, le critère principal concerne la nature économique du produit avec d'un côté des biens tangibles (composants) ou intangibles (progiciels) dont la production s'apparente à une production industrielle, et d'un autre côté des services (logiciels sur mesure ou autres services informatiques).

Le dernier problème de délimitation, qui prend une importance grandissante avec la banalisation de l'utilisation de l'informatique dans les domaines les plus divers, est la séparation, dans un produit, entre ce qui relève d'une activité de programmation (la composante logicielle du produit ou de l'activité) et ce qui relève de la mobilisation de compétences non informatiques. C'est particulièrement net avec le développement de produits multimédias. Par exemple, doit-on considérer que l'encyclopédie Universalis sur CD-ROM est un progiciel en insistant sur les nouvelles fonctionnalités de recherche et de navigation (liens hypermédias) permises par ce type de support, ou considérer qu'il s'agit toujours d'un livre en insistant sur la permanence du contenu par rapport à sa version papier ? De même, la réalisation d'un site Web mêle inextricablement des compétences de graphistes, de

communicateurs et d'écritures de programmes informatiques. La fusion des activités de programmation et des activités non informatiques que réalisent les programmes, est une tendance inéluctable liée à l'éloignement des logiciels du matériel et à la réalisation de fonctions de plus en plus sophistiquées par l'informatique.

B - LE PROGICEL, UN BIEN PARTIELLEMENT COLLECTIF ?

Quand un programme informatique existe sous forme de progiciel, se posent des problèmes de production et de diffusion, spécifiques aux biens intangibles, qui possèdent certaines caractéristiques des biens collectifs. Si ces questions ne concernent a priori que les progiciels et non les logiciels sur mesure, il importe de remarquer que les progiciels constituent une part croissante des logiciels, que les frontières entre ces deux activités ne sont pas étanches et que de plus en plus des logiciels sur mesure seront produits à partir de composants logiciels standardisés, pour lesquels on rencontre les mêmes types de problèmes que pour les progiciels. Après avoir constaté que les progiciels possèdent "naturellement" les déterminants des biens collectifs (1), nous analyserons les mécanismes techniques (2) et juridiques (3) élaborés pour contrôler sa reproduction et en faire un produit commercialisable.

1 - Les caractéristiques des biens collectifs et les progiciels

Les biens collectifs ont des caractéristiques particulières concernant leur production (indivisibilité), leur utilisation (bien non rival) et leur distribution/acquisition (non-exclusion de l'usage).

a - Indivisibilité

Un bien est indivisible quand le coût de production est indépendant du nombre d'utilisateurs. Or un progiciel possède une très grande facilité de reproduction : "créer un nouvel exemplaire à partir d'un programme existant est rapide, peu coûteux et ne nécessite pas de matériel spécialisé puisque le matériel destiné à utiliser le programme permet aussi de le reproduire" (Gérard Dréan, 1996 A, p. 219)[27]. De ce fait, on peut dire que le progiciel se

[27] Pamela Samuelson, experte en droit informatique, fait remarquer que vendre un logiciel ressemble à vendre une voiture avec les clés de l'usine. Lauren Ruth Wiener, qui rapporte cette observation, ajoute qu'il est plus simple de copier une disquette que de faire tourner une usine automobile (1994, p. 105) !

caractérise par une indivisibilité croissante, liée à la baisse du coût des supports sur lesquels il est fourni et dont l'importance a toujours été négligeable par rapport au coût de développement de l'original (baisse du coût des disquettes et CD, documentation sous une forme numérique au détriment de la forme papier…). Pour le producteur, la vente d'une nouvelle unité a même un coût nul quand le progiciel est téléchargé sur Internet, situation qui se développe rapidement. De ce point de vue, on peut dire que la production d'un progiciel s'apparente à la production de connaissances de type scientifique et technique (coûts de production très lourds et coûts de reproduction quasiment nuls).

On peut toutefois nuancer légèrement cette caractérisation, si l'on ne considère pas uniquement le nombre des utilisateurs, mais également leur diversité et les spécificités des matériels qu'ils utilisent. En effet le coût de développement du progiciel original croît significativement quand ce progiciel est destiné à être utilisé par des personnes aux compétences différentes, sur des matériels de plus en plus hétérogènes et en interaction avec d'autres progiciels très divers. Enfin, il est évident que le coût des services de maintenance, d'installation, de formation, parfois fournis avec le progiciel, augmente avec le nombre de progiciels livrés, mais on peut considérer que ces services sont distincts sur le plan de l'analyse économique du progiciel lui-même, comme le prouve le fait qu'ils peuvent être assurés par d'autres prestataires que la société productrice du progiciel.

b - Bien non-rival

Un bien non rival est un bien qui ne se détruit pas dans l'usage et qui peut donc être adopté par un nombre infini d'utilisateurs, sa consommation ou son usage par un agent économique n’empêchant pas sa consommation ou son usage par les autres (indivisibilité dans l'usage). Si l'on considère que ce qui constitue fondamentalement un progiciel c'est "l'original" du produit, il est clair qu'il s'agit d'un bien non-rival. C'est moins évident si l'on prend en compte comme identité économiquement pertinente, un exemplaire particulier inscrit sur un support donné et en général identifié par un numéro de série. En effet, son utilisation par un nouvel agent économique présuppose une opération de copie. Mais outre la facilité de cette opération, le fait que le détenteur du progiciel qui a servi à la copie le détienne toujours après la réalisation de la copie, fait que celle-ci apparaît fréquemment comme légitime, à défaut

d'être toujours légale. Par conséquent, il n'existe une spoliation (limitée) que par rapport au créateur de l'original, ce qui fait qu'elle est considérée très différemment, d'un point de vue éthique, de l'appropriation d'un bien (tangible) appartenant à autrui[28].

c - La non-exclusion de l'usage

Un bien est appropriable s'il est possible pour celui qui l'utilise ou le consomme d'exclure tout autre utilisateur ou consommateur potentiel. Cette propriété d'*excludability*, que Michel Callon (1994 A, p. 6) propose de traduire par "exclusivisme", n'existe pas a priori pour un logiciel, dont il est difficile d'empêcher l'utilisation par un agent économique. Les progiciels, comme les autres biens intangibles, se caractérisent par leur fluidité. Leur production n'est pas limitée par des contraintes physiques, leur consommation n'est pas soustractive et l'on peut parler à leur propos d'économie d'abondance (Charles Goldfinger, 1994, p.14-15).

Toutefois, il existe des possibilités techniques ou juridiques de rendre un progiciel appropriable, permettant de le transformer en un produit commercialisable. Il est intéressant de noter que les efforts[29] portent sur le rétablissement artificiel d'une situation de rareté au contraire de l'activité économique "normale" qui est de lutter contre celle-ci[30], selon une logique qui n'est pas sans rappeler les tentatives de création de semences qui ne peuvent se reproduire.

2 - Les restrictions du caractère collectif par des procédés techniques

Deux solutions existent sur un plan technique : la transformation en un bien tangible et l'adjonction d'un dispositif technique empêchant la reproduction du progiciel sans l'accord de la société productrice.

---

[28] Un constat identique peut être fait pour le photocopiage des livres et plus généralement pour la reproduction "sauvage" des biens intangibles.

[29] En analysant de façon plus globale les dimensions de bien public de la connaissance, Sandra Braman note que "les efforts déployés pour trouver des moyens diversifiés de transformer cette connaissance en propriété privée représentent un des plus âpres combats de cette décennie" (1997, p. 99).

[30] "Un produit vendable se trouve dans une situation de rareté relative. Le logiciel est-il une ressource rare ? En un certain sens oui, mais cette rareté s'entretient artificiellement par des mécanismes juridiques. Elle n'est pas naturelle" (Bernard Lang, 1998 C).

a - Par la transformation en biens tangibles

La première solution, la plus radicale, consiste à graver le programme dans le silicium sous forme d'un composant électronique ROM. Il s'agit dès lors d'un bien tangible qui ne possède plus aucune caractéristique des biens collectifs. Cette solution qu'avait envisagé Bill Gates lors de la production de son premier progiciel (un interpréteur BASIC pour micro-ordinateur) face à la recopie des cartes perforées sur lequel il était stocké[31], constitue une protection efficace contre la copie, dans la mesure où celle-ci nécessite des compétences, du matériel spécialisé et est coûteuse[32], mais elle présente beaucoup d'inconvénients qui font qu'elle est peu utilisée en pratique. Tout d'abord, elle est difficilement applicable pour des programmes de taille importante. Ensuite, même quand il est produit en grande série la production d'une unité supplémentaire a un coût non négligeable, ce qui rendrait ce programme peu compétitif par rapport à un programme concurrent commercialisé sous forme de progiciel. Enfin et surtout, à la différence d'un logiciel qui peut être modifié très facilement, un composant électronique se caractérise par sa rigidité ce qui pose deux problèmes majeurs : premièrement, la correction des défauts initiaux, inévitables dès qu'un programme atteint un certain niveau de complexité, nécessite de détruire les composants défectueux et d'en produire de nouveaux, alors qu'il est aisé de distribuer une nouvelle version voire un simple additif logiciel (*patche*) dans le cas d'un progiciel[33]. Deuxièmement, il est plus difficile de produire et de faire accepter des nouvelles versions d'un composant électronique que d'un progiciel, dans un secteur où l'obsolescence programmée est une source de revenus importante.

b - Par l’adjonction de dispositifs techniques

La deuxième solution consiste à adjoindre un dispositif technique visant à empêcher la copie du progiciel (le dispositif est présent sur le support de stockage du progiciel) ou à lui retirer tout intérêt (le progiciel ne peut fonctionner qu'avec la présence sur le matériel d'un

[31] Du reste, certains micro-ordinateurs commercialisés au début des années quatre-vingt ont comporté un interpréteur BASIC stocké dans la mémoire ROM.

[32] Cette efficacité est toutefois relative à l'évolution des techniques : la reproduction d'un CD-ROM, opération difficile et coûteuse il y a quelques années, est devenue un jeu d’enfant (dans tous les sens du terme) actuellement.

[33] Une illustration de cette différence est fournie par les conséquences économiques importantes pour la société productrice (Intel) du constat exceptionnel d'un défaut dans la programmation du microprocesseur Pentium II, comparativement à l'existence habituelle de défauts dans la plupart des progiciels commercialisés qui n'affectent nullement les résultats des sociétés productrices.

composant difficilement reproductible livré avec le progiciel). L'efficacité de ces dispositifs est limitée : développement de logiciels permettant de copier des disquettes même protégées[34], modification du progiciel pour contourner la vérification de la présence du composant électronique. En effet à la différence de dispositifs de protections voisins comme les décodeurs de télévision, il n'est pas forcément nécessaire de reproduire le dispositif matériel qui peut toujours être neutralisé par une opération logicielle. Certes cette activité peut être très complexe mais elle a toujours suscité une inventivité débordante de la part de *hackers* opposés à la rétention de la circulation d'une information technique (Philippe Breton, 1990, p. 111). Leur motivation réside principalement dans la réalisation d'un exploit technique (d'où l'inefficacité de la sophistication des protections techniques)[35], les progiciels "déprotégés" étant rarement revendus, voire même fréquemment non utilisés par ceux qui ont effectué l'opération. Par contre leur diffusion est rapide et il n'est pas rare de voir apparaître une version "déprotégée" avant même la commercialisation du progiciel considéré, phénomène qui ne peut que s'étendre avec le développement d'Internet[36].

Par contre ces formes de protection peuvent rendre le progiciel moins efficace, être la source de complications et surtout constituer des difficultés d'utilisation (pouvant aller jusqu'à l'impossibilité d'utilisation) pour des utilisateurs "légaux". C'est pour ces raisons que ces dispositifs sont de moins en moins présents sauf pour certains progiciels particulièrement coûteux et destinés à un public spécialisé.

Aucune solution de nature purement technique ne répondant correctement aux exigences de protection des progiciels (Jean-Benoît Zimmermann, 1995 B, p. 182), c'est sur le terrain juridique que vont porter la plupart des efforts pour garantir "l'appropriabilité" des progiciels.

---

[34] Ces logiciels sont même, dans certains cas, commercialisés tout à fait légalement, sous le prétexte de permettre de réaliser une copie de sauvegarde.

[35] Une association célèbre de *crackeurs* ("pirates informatiques") est l'association d00dz (*day zero*) dont l'objectif est de forcer les sécurités des logiciels de jeu, le jour même de leur sortie officielle.

[36] Le même phénomène se reproduit actuellement pour les œuvres musicales où le format MP3 de transfert audio sur Internet permet une reproduction facile des œuvres. Des tentatives existent pour promouvoir d'autres standards et les logiciels correspondants, de façon à limiter les copies (en nombre et dans le temps) mais ces tentatives sont régulièrement détournées par des *hackers* : le dernier exemple est le logiciel multimédia de Microsoft, Windows Media Audio 4.0, qui devait permettre d'imposer le respect des droits d'auteur en limitant la gravure à un CD sur une période limitée (la copie s'altérant automatiquement au bout d'un certain temps) ; ses protections ont réussi à être contournées deux jours après l'annonce de son lancement (01 Informatique, 20/08/1998).

3 - Les restrictions du caractère collectif par des procédés juridiques et la multiplicité des situations existantes

a - L'importance de la question des droits de propriété intellectuelle dans l'économie du logiciel

L'ampleur du '"piratage" ou copie illicite de logiciel montre l'importance de cette question. La Business Software Alliance (BSA) calcule un "taux de piratage" qui est le rapport entre les logiciels d'application installés (demande) et les expéditions licites de logiciels d'application (offre).

Tableau VIII

Average Software Piracy Rates by Region, 1994–1998
(percentage of applications installed each year)

Region	1994	1995	1996	1997	1998
Asia/Pacific	68	64	55	52	49
NorthAmerica	32	27	28	28	26
United States	31	26	27	27	25
Western Europe	52	49	43	39	37
Latin America	78	76	68	62	60
Eastern Europe	85	83	80	77	76
Middle East/Africa	80	78	74	65	62
World total	**49**	**46**	**43**	**40**	**38**

Source : Business Software Alliance, 1999, p. 22.

Il importe toutefois de noter qu'il ne s'agit que d'une estimation dont la fiabilité, comme pour toute mesure d'un phénomène illégal, est difficile à apprécier[37]. De plus elle émane d'un regroupement des grands acteurs informatiques américains, dont l'intérêt est de souligner l'ampleur du phénomène pour susciter une répression accrue. Dans son rapport de 1999, la

[37] Certains indices indirects sont toutefois éloquents : en Russie un livre sur le logiciel Clipper s'est vendu à 1 193 000 exemplaires alors que Nantucket, l'éditeur de ce logiciel, n'avait enregistré que trois commandes de son produit (Frédéric Dromby, 1999, p. 645).

BSA estime que le piratage informatique aurait ainsi coûté 109 000 emplois à un secteur qui compte 809 000 salariés et aurait représenté un manque à gagner fiscal de près de 1 milliard de dollars pour l'administration américaine (Business Software Alliance, 1999). La BSA calcule également des pertes de chiffre d'affaires dues au piratage des logiciels :

Tableau IX
Pertes de chiffre d'affaires dues au piratage des logiciels
en milliards de dollars

	1994	1996
Europe	3,9	3,4
Asie / Pacifique	3,1	3,7
Amérique du Nord	3,9	2,7
Amérique Latine	1,0	0,9
Moyen-Orient / Afrique	0,4	0,5

Source : BSA / SPA Global Software Piracy Report, Facts and Figures, 1994-96

Indépendamment de l'importance réelle du piratage, le manque à gagner, estimé par le prix de vente des logiciels piratés, est vraisemblablement surestimé, si l'on considère que de nombreuses sociétés n'auraient pas acheté ces logiciels si elles avaient été contraintes de se les procurer légalement. Il faut de plus ajouter que le piratage d'un progiciel peut aider à la diffusion de sa version commerciale, ce qu'illustre le constat que les logiciels les plus copiés sont également les plus vendus ou les stratégies de fourniture gratuite de certains progiciels pour les imposer sur le marché.

La répression du piratage a tendance à se durcir et est même devenue un sujet de conflit entre certains pays (par exemple entre les Etats-Unis et la Chine où le taux de piratage est estimé à 90 %), ce qui pourrait expliquer la baisse constatée des taux de piratage et du manque à gagner pour les éditeurs. On peut citer l'exemple d'une agence de publicité brésilienne Artplan, poursuivie par la *Business Software Alliance* (BSA), qui a été condamnée à verser une amende de 65 millions de dollars pour l'utilisation frauduleuse de 382 programmes, soit 170 000 dollars le logiciel (Jean-Paul Smets-Solanes, Benoît Faucon, 1999, p. 132). Par contre, la vente de progiciels en ligne par Internet pose de nouveaux problèmes : le Software

& Information Industry Association (SIIA) a effectué une étude entre le 15 et le 20 août sur les logiciels vendus aux enchères sur Internet par de grands sites (eBay, ZDNet.com et Excite.com) d'où il ressort que 60 % étaient des logiciels pirates ; l'étude cite l'exemple de Macromedia Director, un logiciel très utilisé par les graphistes, dont le prix de vente est de 999 dollars, qui était vendu aux enchères au prix de 28 dollars (Les News, 9/9/99).

En plus des problèmes de piratage se pose le problème de l'imitation ("clonage"), qui n'est certes pas propre au progiciel, mais qui a pris une dimension importante dont témoigne le nombre de procès[38] et la vigueur des actions entreprises : en 1990, des *hackers* ont été arrêtés par le FBI pour avoir divulgué des extraits du code source du système d'exploitation d'Apple.

Plus généralement pour le secteur du logiciel, "les droits de propriété intellectuelle apparaissent comme une variable clé au moins aussi importante que la capacité productive ou le facteur travail" (Robert Merges, 1996, p. 285). Cet auteur montre comment cette question peut avoir une importance décisive dans la stratégie de l'entreprise et cite l'exemple de l'acquisition d'Asthon-Tate (éditeur de dBase, produit leader à l'époque dans les systèmes de gestion de bases de données pour micro-ordinateurs) par Borland en 1991 ; le Département de la Justice autorisa l'opération à la condition que Borland renonce à des droits de propriété "forts" sur dBase et abandonne un procès contre Fox (un autre éditeur de SGBD). En quelques années Microsoft, qui avait racheté Fox, devint hégémonique sur ce segment de marché avec un produit Access "compatible" avec dBase.

Mais si les questions de droits de propriété sont importantes dans ce secteur, elles sont également très complexes en raison de la nature particulière du logiciel.

b - Une question complexe vu la nature particulière du logiciel

La question du statut juridique du progiciel est particulièrement délicate car elle mêle indissociablement deux problèmes. Le premier problème, qui n'est pas spécifique au progiciel, concerne le degré socialement optimal de la protection. Il correspond à un dilemme, qualifié de *schumpeterien*, concernant l'influence du régime des droits de propriété sur le dynamisme technologique, la protection devant tout à la fois constituer un facteur incitatif suffisant *ex ante* sans trop entraver l'imitation et la diffusion *ex post* (Jean-Alain Héraud, 1995, p. 91). De

ce fait, le débat a toujours été vif entre partisans d'une protection "forte", qui permet d'accroître les incitations à innover, et partisans d'une protection "faible" insistant sur les bienfaits d'une diffusion facilitée des innovations pour élargir la concurrence et éviter les pertes d'efficience liées aux effets de duplication de la recherche (Partha Dasgupta, Joseph Stiglitz, 1980).

A ce premier problème s'ajoute la question de la nature particulière du logiciel comme texte numérique actif (cf. chapitre 1), le fait que "le logiciel est à la fois technologie et expression" (Jean-Benoît Zimmermann, 1995 B, p. 182).

En tant qu'expression intellectuelle, un logiciel, est une prestation intellectuelle, assimilable à une œuvre de l'esprit au même titre qu'une œuvre littéraire ou artistique, propriété incorporelle exclusive de son auteur, et semble devoir relever de la législation sur les droits d'auteur, le support matériel sur lequel repose le logiciel n'étant considéré que comme un accessoire à la prestation intellectuelle.

Mais un logiciel est également une technologie, "un processus (…) pour réaliser une tâche de calcul, de traitement et de manipulation de données, de commandes" (Jean-Benoît Zimmermann, 1995 B, p. 183), un procédé permettant de tirer parti des ressources du matériel en vue d'un résultat déterminé. A ce titre, il peut également prétendre à une protection par le brevet d'invention qui permet l'exercice d'un droit de monopole sur un dispositif ou une méthode, consenti selon les critères de nouveauté, d'originalité et de non-évidence. En définitive, le logiciel ne peut "trouver commodément sa place ni dans l'une, ni dans l'autre de ces deux branches du droit de la propriété intellectuelle, ce qui fait toute la difficulté en la matière" (André Lucas, 1987, p. 195). Ces difficultés sont une illustration particulière de "la crise qui frappe aujourd'hui les systèmes de droit de propriété intellectuelle" (Dominique Foray, 1995, p. 136), le logiciel étant "un « cas d'espèce » qui comporte une signification prospective pour l'ensemble de l'industrie" (Jean-Benoît Zimmermann, 1995 B, p. 182).

Dans le cas du logiciel, il faut de plus prendre en compte les aspects très différents qu'il intègre, et qui ne disposent pas tous du même type et du même niveau de protection : la résolution non informatique du problème (pour un logiciel applicatif) - qui fréquemment

[38] Par exemple, Xerox a intenté des procès contre tous les éditeurs de logiciels qui développent des interfaces graphiques pour lesquelles elle s'estime pionnière ; pour les mêmes raisons Apple a attaqué Hewlett-Packard et Microsoft.

reposait en partie sur des savoir-faire, mais qui en étant transformée en connaissances codifiées devient facilement diffusable -, les algorithmes informatiques utilisés, leur mise en application concrète pour résoudre le problème considéré, le *look and feel*[39] de l'application, les interfaces avec le matériel et les autres programmes.

D'une façon générale, après d'intenses débats dans les années soixante-dix, les différents Etats ont renoncé à adopter un cadre juridique spécifique pour le logiciel compte tenu de l'ampleur et de la difficulté de la tâche, et les cadres existants (droits d'auteur et brevets) ont simplement été adaptés (Jean-Benoît Zimmermann, 1995 B, p. 188). Au départ, c'est essentiellement le droit d'auteur qui a été mobilisé, avec des modalités différentes selon les pays, mais de plus en plus se développent des tendances à la brevetabilité des logiciels.

c - Des solutions différentes dans le temps et dans l'espace

Pendant longtemps en France, les logiciels n'étaient pas considérés par la jurisprudence comme relevant de la loi sur la propriété des œuvres littéraires ou artistiques (loi du 11 mars 1957). Ce n'est que le 2 novembre 1982 qu'un jugement de la quatrième chambre de la cour d'appel de Paris a accordé un "droit d'auteur" à un informaticien pour un programme d'ordinateur qu'il avait conçu. A la suite d'une série d'arrêts contradictoires, le législateur a adopté une loi (3 juillet 1985) qui étend explicitement à la protection des programmes informatiques le champ de la loi du 11 mars 1957. Cette loi sera complétée par la loi du 10 mai 1994 qui parle pour la première fois de logiciel et plus seulement de programme d'ordinateur. Ces lois excluent clairement de la protection par droit d'auteur les principes et algorithmes, conformément à la jurisprudence française sur la propriété intellectuelle qui précise que les œuvres protégées doivent être cristallisées dans une forme, les idées n'étant pas protégées par le droit d'auteur.

Sur la question de la brevetabilité des logiciels il s'est produit une certaine évolution. La loi sur les brevets d'invention du 2 janvier 1968 exclut explicitement les "programmes ou séries d'instructions pour le déroulement des opérations d'une machine calculatrice" (article 7). Cette position était motivée essentiellement par la crainte d'asseoir sur le long terme la position dominante d'IBM (Jean-Benoît Zimmermann, 1995 B, p. 190). Cette restriction sera confirmée par la loi du 13 juillet 1978 qui visait à harmoniser la législation française avec le

[39] Le *look and feel* est, pour l'utilisateur, l'aspect externe du programme, défini notamment par l'interface-

texte de la Convention de Munich. Toutefois la jurisprudence très stricte au départ a admis comme brevetable les logiciels présentés comme parties intégrantes et inséparables de processus industriels : "un programme informatique est indirectement brevetable aux conditions de fond suivantes ; il doit constituer une étape d'élaboration d'un procédé industriel brevetable ; il doit satisfaire lui-même aux trois critères de brevetabilité : la nouveauté, le résultat industriel et l'activité inventive" (Cour d'Appel de Paris, affaire Schlumberger, 15-6-81). A l'heure actuelle, ces questions relèvent de plus en plus de la juridiction européenne et c'est au niveau de la Communauté Européenne qu'est débattue la brevetabilité des logiciels.

En Europe

Au niveau européen, une Directive votée par le parlement le 17 avril 1991, adoptée à l'unanimité par le Conseil des Ministres les 12 et 14 mai, est entrée en application pour tous les pays membres le premier janvier 1993. Cette directive définit un droit du logiciel fondé sur la protection du contenu d'un logiciel par le droit d'auteur. Elle exclut du champ de la protection la logique, les algorithmes et la langue de programmation. Elle protège "l'écriture" d'un programme tout en laissant la liberté à d'autres auteurs d'écrire un logiciel différent pour arriver à un même résultat. Elle autorise uniquement la copie de sauvegarde, la correction d'erreurs dans le cadre de l'utilisation prévue et, sous certaines conditions, la décompilation dans une perspective d'interopérabilité des programmes entre eux et avec les ordinateurs (si les éditeurs ne communiquent pas les informations nécessaires à la réalisation d'interfaces).

Par ailleurs, l'article 52-2 c de la Convention de Munich (1973) interdit de breveter un logiciel. Toutefois, après avoir exclu par principe les logiciels de la brevetabilité, l'Office Européen des Brevets a nuancé sa position : en se basant sur le fait qu'un brevet d'invention est délivré pour toute invention nouvelle, impliquant une activité inventive et susceptible d'application industrielle, les chambres de recours de l'Office Européen des Brevets ont considéré un logiciel comme une invention "s'il engendre un effet technique qui va au-delà des interactions physiques normales existant entre un programme d'ordinateur et un ordinateur". "Si l'objet revendiqué apporte une contribution de caractère technique, la brevetabilité ne devrait pas être mise en cause pour la simple raison qu'un programme d'ordinateur est impliqué" (directive donnée aux examinateurs de l'OEB). En théorie c'est le procédé impliquant le logiciel qui est brevetable, pas le logiciel en tant que tel. En fait, selon

utilisateur, la présentation des commandes…

Pierre Breese, avocat conseil en propriété industrielle, il suffit de faire passer le logiciel comme "procédé" pour que le brevet soit accepté. Pierre Breese estime qu'il y a aujourd'hui 10 000 brevets européens portant objectivement sur des logiciels (ZDNet, 23-6-99). A l'occasion de la révision de la Convention de Munich, les éditeurs de progiciels ont tenté d'élargir encore les possibilités de breveter des logiciels, ce qui a suscité de vigoureuses oppositions notamment des partisans des logiciels libres. La conférence intergouvernementale européenne du 25 juin 1999 a repoussé à plus tard la décision et a mis en place un groupe de travail chargé de "clarifier la situation" (ZDNet, 30-6-99).

Au Japon

Au Japon la protection de la propriété intellectuelle est en général assez faible. C'est sous la pression des poursuites contre des entreprises japonaises pour violation de droits de propriété relatifs à des logiciels[40], qu'une intégration du logiciel à la loi sur le copyright sera effectuée en 1985. La nouvelle loi japonaise exclut toutefois de la protection par copyright les langages de programmation, les algorithmes et les interfaces *(look and feel)*, et elle maintient le droit à la modification des programmes pour en améliorer l'efficience ou pour les adapter à un système non compatible. La décompilation est autorisée sauf pour les logiciels américains à la suite d'un accord bilatéral avec les Etats-Unis (Jean-Paul Smets-Solanes, Benoît Faucon, 1999, p. 74). Il existe toutefois peu de jurisprudence et beaucoup d'incertitudes qui font que la protection par copyright connaît des limites significatives (Robert Merges, 1996, p. 275).

Par contre le système japonais de brevets qui combine droits de propriété faibles, exigences de nouveauté peu élevée et contrainte forte de divulgation précoce (Dominique Foray, 1995, p. 142) se traduit par un nombre élevé de dépôts de brevets sur des procédés liés au logiciel, estimé à environ 35 000 chaque année (Jean-Paul Smets-Solanes, Benoît Faucon, 1999, p. 141).

Aux Etats-Unis

[40] L'affaire la plus célèbre est le procès intenté par IBM à Hitachi et Fujitsu pour utilisation illicite de son système d'exploitation grand système MVS/XA.

Aux Etats-Unis, suite aux travaux d'une commission mandatée par le Congrès, la CONTU (*National Commission on New Technolgical Uses of Copyrighted Works*), le logiciel a été pris en compte dans le champ juridique du copyright (*Computer Software Copyright Act* de 1980 qui amendait le *Copyright Act* de 1976). L'application de cette décision à certains aspects particuliers des logiciels a posé des problèmes juridiques complexes. Le *look and feel* d'un logiciel n'a pas de statut juridique précis mais la jurisprudence tend à le faire bénéficier de la protection par copyright depuis que Lotus a obtenu gain de cause contre Paperback en juin 1989 (Jean-Benoît Zimmermann, 1995 B, p. 194). Les interfaces d'un logiciel ont fait l'objet d'interprétations contradictoires mais il semble que se dégage un consensus pour les exclure du droit d'auteur (Jean-Paul Smets-Solanes, Benoît Faucon, 1999, p. 76). Les opérations de décompilation, y compris pour produire des logiciels compatibles, ont été considérées comme une violation du droit d'auteur dans un procès célèbre et qui avaient mobilisé de nombreux acteurs de l'informatique (Sega contre Accolade en avril 1992) et sont interdites à de rares exceptions près. La protection par droit d'auteur de la structure des programmes fait l'objet de jurisprudences contradictoires. Il faut toutefois nuancer l'efficacité de la protection par copyright. En cas d'accusation de violation de copyright, beaucoup de sociétés ripostent en invoquant l'utilisation du copyright à des fins non concurrentielles, et notamment qui contreviendrait aux lois antitrust (Robert Merges, 1996, p. 285).

Concernant les possibilités de breveter des logiciels c'est aux Etats-Unis que l'évolution a été la plus importante. Alors que les copyrights dépendent d'un organe législatif (la Librairie du Congrès), les brevets émanent d'un service gouvernemental relevant du Département du Commerce. La position de départ du *Patent and Trademark Office* (TPO) était hostile à la délivrance de brevets pour des programmes d'ordinateurs, en raison de leur nature assimilable à un enchaînement d'étapes abstraites et/ou d'algorithmes mathématiques. Jusqu'au début des années quatre-vingt la Cour Suprême, en s'en tenant au principe de lier les algorithmes informatiques à des "lois de la nature" et à des principes scientifiques, refusait les brevets sur les logiciels et allait jusqu'à invalider certains brevets. Mais le changement de la perception des problèmes des droits de propriété en général et le fait que la Cour Suprême ait cédé la juridiction à la *Court of Appeals for the Federal Circuit*, créée en 1982 et favorable aux

brevets, ont modifié la situation[41] (Robert Merges, 1996, p. 280). A partir des années quatre-vingt le nombre des brevets accordés croit très fortement. Jean-Paul Smets-Solanes et Benoît Faucon estiment que le TPO accepte environ 20 000 brevets sur des procédés liés au logiciel chaque année (à titre de comparaison 70 000 brevets sont déposés en France par an tous domaines confondus). A elle seule IBM a déposé 1 087 brevets aux Etats-Unis en 1993, dont 20 % concernaient des logiciels (01 Informatique, n° 1444, 14-3-97). Ces brevets concernent des processus contrôlés par ordinateurs, des méthodes de gestion implantées sur ordinateurs, des interfaces homme-machine, et des algorithmes implémentés par ordinateur (Robert Merges, 1996, p. 279). En 1994 pour la première fois un brevet a été utilisé dans un procès : Microsoft s'est vu condamnée pour violation d'un brevet de compression de données détenu par Stac Electronics avec laquelle elle avait échoué sur les conditions d'un accord de licence. Microsoft a été contrainte à verser 120 millions de dollars de dommages et intérêts à Stac Electronics et à retirer provisoirement le produit en cause du marché.

d - Les problèmes de la brevetabilité des logiciels

La brevetabilité croissante des logiciels pose une série de problèmes. Théoriquement le brevet protège l'invention, une invention étant une solution nouvelle à un problème technique (pas nécessairement nouveau) qui doit satisfaire trois critères : nouveauté, inventivité et applicabilité industrielle (Jean-Alain Héraud, 1995, p. 92). L'invention est distincte de l'innovation et de la découverte scientifique qui n'est pas brevetable (non appropriable et diffusion libre). Les distinctions entre ces trois notions, déjà peu évidentes en général, sont particulièrement problématiques dans le cas des logiciels, où d'une part certains résultats de la recherche débouchent de façon quasi immédiate sur des produits opérationnels (les algorithmes de compression de données et de cryptage par exemple), et où l'évolution technologique résulte essentiellement d'un processus continu d'innovations incrémentales.

Mis à part les applications des découvertes de la recherche fondamentale, les véritables inventions sont souvent le fruit d'une idée nouvelle concernant un produit, que ce soit le produit lui-même (tableur, navigateur), ses interfaces (souris, interface graphique) ou des

[41] Par exemple en 1982, le TPO avait rejeté une demande de brevet sur un système de reconnaissance du signal basé sur un nouvel algorithme. Cette demande sera acceptée par la Court of Appeals for the Federal Circuit en 1989. Cette dernière reconnaissait que l'invention concernait un algorithme, *a priori* non brevetable, mais que le brevet, accordé au programme résidant en ROM, n'empêchait pas l'utilisation de l'algorithme qui reste dans le domaine public (Robert Merges, 1996, p. 280).

fonctions nouvelles[42]. Dans ces situations où l'invention réside plus dans l'idée elle-même que dans sa mise en œuvre, le droit d'auteur qui protège seulement l'expression originale d'une idée n'est pas adapté. Mais dans les cas où les inventeurs ont essayé d'utiliser la protection par le brevet, celle-ci ne s'est pas révélée non plus efficace, l'exemple le plus célèbre étant la tentative de Xerox de faire breveter ses inventions concernant la souris et les interfaces graphiques : le brevet a été annulé par la justice américaine, qui a considéré qu'il s'agissait d'une idée et non d'un procédé, et a été accordé à Apple qui s'était pourtant fortement inspirée des travaux de Xerox dans ce domaine (Jean-Paul Smets-Solanes, Benoît Faucon, 1999, p. 173). Dans ce domaine, les progiciels apparaissent comme une bonne illustration du modèle de David Teece (1986) où l'innovateur tire moins de bénéfices de son innovation qu'un imitateur disposant d'actifs complémentaires des savoir-faire d'innovation (circuits de financement, de distribution…). Par exemple, dans les progiciels bureautiques pour micro-ordinateurs Microsoft est devenu progressivement hégémonique et les sociétés qui avaient développé le premier tableur (Visicalc), le premier traitement de texte (Wordstar) et le premier système de gestion de bases de données (dBase) ont toutes disparu.

Les brevets pourraient sembler plus adaptés aux inventions concernant des procédés. Mais dans ce cas, en contrepartie du monopole temporaire d'exploitation qu'il confère à son détenteur, le dépôt d'un brevet nécessite de livrer des informations et fournit des indications aux autres acteurs sur la "praticabilité" d'une direction de recherche, ses potentialités (Michel Callon, 1994 A, p. 8)[43]. De plus, dans le cas des logiciels, les améliorations résultent essentiellement de l'application de principes généraux (donc non brevetables) et d'un flux continu d'améliorations incrémentales.

---

[42] Elles correspondent à ce que André Barcet, Joël Bonamy et Anne Mayère (1987) appellent des innovations fonctionnelles (créer ou répondre à de nouveaux besoins) et dont ils ont montré l'importance dans le cas plus général des services intellectuels, ainsi que les difficultés d'appropriation et de protection.

[43] Cet aspect des brevets qui apparaissait comme un mal nécessaire à l'origine est de plus en plus considéré comme un avantage pour la diffusion de l'innovation comparativement à la pratique du secret (Dominique Foray, 1995, p. 139). Toutefois il peut être annulé par des pratiques de multiplication extrême de dépôts de brevets, ne correspondant pas à des perspectives économiquement intéressantes, mais visant à dissimuler la stratégie de recherche et à orienter les imitateurs potentiels vers de fausses pistes (Jean-Alain Héraud, 1995, p. 11).

De ce fait la plupart des brevets déposés concernent des questions relativement mineures[44], voire triviales[45] ou conventionnelles[46]. Certes en théorie, à l'exigence d'originalité définie par la législation du copyright s'ajoute celle de la nouveauté dans le cadre du système de brevet. Mais le critère de nouveauté est difficile à fonder en raison de l'absence d'un inventaire de l'état de l'art et des pratiques antérieures de dépôts de brevets (Jean-Benoît Zimmermann, 1995 B, p. 193). Les délais d'application (32 mois en moyenne) sont en totale dysharmonie avec le cycle de vie très bref du produit (Dominique Foray, 1995, p. 139). Aux Etats-Unis, le dépôt des brevets sur les logiciels est devenu si important que le bureau américain des brevets est devenu incapable de procéder à des recherches d'antériorité fiables : par exemple, un brevet a été récemment accordé à Microsoft pour une technologie de lissage de caractères déjà publiée en 1976 (Jean-Paul Smets-Solanes, Benoît Faucon, 1999, p. 170).

La plupart des grandes sociétés de logiciels, qui disposent de moyens financiers importants, déposent de multiples brevets[47] dont la validité est douteuse. De toute façon, la plupart des plaintes pour contrefaçon se résolvent par des échanges de brevets et des accords

---

[44] On peut citer aux Etats-Unis, le brevet sur la technique *natural order recalc* qui permet de recalculer les conséquences sur l'ensemble d'une feuille de calcul d'une modification du contenu d'une cellule ; ce principe utilisé dans tous les tableurs fait l'objet par le détenteur du brevet d'un procès contre six éditeurs de progiciels (Jean-Benoît Zimmermann, 1995, p. 192). Autre exemple, le brevet n° US5860073 accordé le 12 janvier 1999 à la société Microsoft par l'office américain des brevets, sur "la feuille de style dans un document électronique" dont la contribution technique réside dans le fait que "contrairement aux autres systèmes, on peut définir le style d'une partie du document avant d'y avoir entré le texte" (Philippe Riviere, 1999, p. 23) ; de plus, Microsoft avait fait adopter auparavant comme standard par le World Wide Web Consortium (W3C) cette technologie de mise en page (Jean-Paul Smets-Solanes, Benoît Faucon, 1999, p. 169).

[45] Jean-Paul Smets-Solanes et Benoît Faucon (1999, p. 141) citent l'exemple de la fonction mathématique "ou exclusif" qui a été brevetée pour son application au procédé d'inversion des couleurs sur un écran (exemple du clignotement du curseur) et qui voit des sociétés attaquées pour contrefaçon. Ce procédé est tellement connu et élémentaire que personne n'avait osé déposer un brevet auparavant. Les mêmes auteurs mentionnent également le principe élémentaire consistant à vendre de la musique téléchargeable sur Internet qui fait l'objet d'un brevet aux Etats-Unis (p. 169).

[46] Un exemple extrême de demande de brevet, toutefois repoussé, est le cas de Lotus qui voulait protéger la combinaison de touches "/ " et une lettre pour ouvrir un menu (par exemple "/F" pour le menu File) (Robert Merges, 1996, p. 289).

[47] Pour la septième année consécutive, IBM est la société qui a déposé le plus de brevets (2756 en 1999), soit 900 de plus que son plus proche concurrent (Canon). Le portefeuille de brevets d'IBM est estimé à 30 milliards de dollars et génère des revenus de plus d'un milliard de dollar par an. Ces brevets concernent des composants matériels (par exemple les semi-conducteurs) et de plus en plus fréquemment des aspects liés à des logiciels. En 1999, ce sont surtout des questions liées au commerce électronique (méthode d'amélioration des transactions en ligne à base d'agents intelligents, système de commande en ligne à partir d'un catalogue électronique…) qui ont fait l'objet de dépôts de brevets de la part d'IBM (01 Informatique, 13-1-2000).

croisés (Robert Merges, 1996, p. 286). Cette pratique de dépôts de brevets pour pouvoir négocier des arrangements avec les concurrents en position de force se rencontre également dans l'industrie automobile (Jean-Alain Héraud, 1995, p. 114). Mais à la différence de l'industrie automobile, le secteur des progiciels se caractérise par la présence à côté de groupes puissants d'une multitude de petites sociétés. Celles-ci n'ont souvent pas les moyens de déposer de nombreux brevets[48] et de contester en justice la validité des brevets qui leur sont opposés : par exemple, une *start-up* française a du cesser ses activités aux Etats-Unis après avoir été attaquée pour contrefaçon ; la société Adobe, attaquée ensuite pour le même brevet, a réussi à le faire annuler lors du procès (Jean-Paul Smets-Solanes, Benoît Faucon, 1999, p. 142)[49].

De fait, "un progiciel moderne peut contenir plusieurs milliers de traitements brevetables individuellement et constituant chacun un risque potentiel de contrefaçon d'un brevet en cours d'examen" (B. Kahin, 1990, p. 40). Enfin, la croissance du nombre de brevets sur des aspects mineurs, peut entraver le développement jugé très prometteur de la production de progiciels à partir de composants provenant de différentes sources, par l'incertitude sur l'étendue du droit de propriété existant sur tel ou tel composant (OTA, 1992, p. 154) [50].

Enfin, le développement de la brevetabilité du logiciel a pris à contre-pied ceux qui entendaient s'appuyer sur une utilisation particulière du droit d'auteur avec la création de licences de logiciels libres (cf. paragraphe suivant) pour limiter au maximum les possibilités d'appropriabilité du logiciel. Contre l'argumentation d'une protection forte favorisant l'effort inventif par l'internalisation de ses effets et par la nécessité d'entrer sur de nouveaux domaines pour échapper à ce qui est protégé (Robert Merges, 1996, p. 286), les promoteurs des logiciels libres argumentent qu'une appropriabilité plus faible peut entraîner un développement

---

[48] Selon Jean-Paul Smets-Solanes et Benoît Faucon, déposer un brevet international coûte près de 100 000 F par an et le défendre en cas de violation coûte plus de 250 000 F (1999, p. 169).

[49] On peut citer également l'exemple de Lotus qui a engagé des actions en justice sur le *look and feel* de son tableur en 1987 contre deux petits éditeurs (Paperback Software et Mosaic Software) et en 1990 contre Borland, mais qui ne s'est jamais attaquée à Microsoft ou à Computer Associates, auxquels elle pourrait faire les mêmes reproches mais qui sont beaucoup plus puissants (Frédéric Dromby, 1999, p. 285 et 578-579).

[50] Ce problème risque de se développer également pour les droits d'auteurs des produits multimédias comprenant de multiples fragments d'œuvres de nature très diverses (textuelles, graphiques, audiovisuelles…).

technologique plus rapide : disparition du coût des droits de propriété, facilité d'entrée et existence de technologies multiples et rivales (idem, p. 290). Si les incitations pour la recherche-développement sont plus faibles, ce phénomène serait plus que compensé par l'importance des *spillovers* (effets de report) qui permet à une entreprise d'accéder aux résultats de la recherche-développement de ses concurrents, ce qu'a mis en évidence le modèle de M. Spence (1984)**.**

e - La multiplicité des statuts juridiques des logiciels

Les progiciels commerciaux ont donc des statuts juridiques variables dans le temps, dans l'espace (cf. c) et qui peuvent être différents pour les différents éléments qui les constituent (code, algorithme, *look and feel*, interface…). Il faut de plus ajouter le cas des logiciels sur-mesure (service individualisé non reproductible) auxquels, sur un plan juridique, s'appliquent classiquement les droits des contrats et le secret des affaires, et prendre en compte le phénomène particulier constitué par les logiciels libres, que l'on peut analyser comme une volonté de maintenir les logiciels comme biens collectifs. Les racines de ce phénomène remontent aux premiers temps de l'informatique avec, dans la communauté universitaire, la tradition d'échange bénévole de programmes considérés comme une création de l'esprit qui doit échapper aux circuits marchands (Gérard Dréan, 1996 A, p. 206). Cette tradition s'est perpétrée sous deux aspects différents.

D'une part, au début de la microinformatique, la pratique de copies libres de logiciels au sein de clubs d'utilisateurs, à laquelle s'opposent violemment les éditeurs de progiciels commerciaux pour micro-ordinateurs (notamment Bill Gates en 1976), donnera naissance à la production de logiciels en *freeware* (ou "gratuiciels") qui sont fournis gratuitement aux utilisateurs, et de logiciels en *shareware* (ou "distribuciels") qui sont diffusés librement, l'utilisateur intéressé par le produit étant tenu moralement de verser une contribution, le plus souvent modeste, à l'auteur du logiciel.

D'autre part, se développe la création de logiciels libres, dont la particularité fondamentale est que leur code source est librement accessible, redistribuable et modifiable. En effet, dans le cas des progiciels commerciaux, comme le plus souvent des logiciels en *shareware* et en *freeware*, c'est uniquement le code objet du logiciel qui est fourni à l'utilisateur, sans possibilité de connaître le code source du logiciel. Sur le plan juridique les logiciels libres sont basés sur des licences particulières, dont la plus utilisée est la Licence

Publique Générale (GPL). Cette licence est l'œuvre de la *Free Software Foundation* (FSF), dont l'objectif est de soutenir le développement de logiciels libres. Cette fondation créée en 1985 par Richard Stallman (un informaticien du MIT) se situait dans le prolongement du projet GNU[51] lancé l'année précédente dont l'objectif était de créer un Unix libre de droit face aux différentes versions commerciales d'Unix des différents constructeurs informatiques. Ces licences publiques utilisent à rebours le droit d'auteur[52] pour garantir qu'un logiciel libre le restera : toute personne peut utiliser, distribuer, modifier librement un logiciel sous licence GPL à condition de maintenir libre ce logiciel, c'est à dire de fournir librement le code source[53]. C'est la différence avec un logiciel du domaine public qui n'étant pas couvert par le droit d'auteur peut faire l'objet de dépôt de licence classique[54]. Les licences publiques posent toutefois un problème par rapport au droit européen. En effet, les licences publiques, comme la plupart des contrats de licence pour des logiciels aux Etats-Unis, ne comportent aucune garantie et lèvent toute responsabilité de la part de l'éditeur. Ceci est en contradiction avec une directive européenne du 25 juillet 1985 qui instaure une protection contre les logiciels défectueux, complétée par une directive du 5 avril 1993 concernant les clauses abusives dans les contrats conclus avec les consommateurs. Ces directives ne s'appliquent qu'à la vente de logiciels à des particuliers ou à des professionnels non informaticiens. Elles ont été confirmées en France par le Garde des Sceaux qui a précisé en août 1998 que "la responsabilité du fait des produits défectueux a vocation à englober la catégorie juridique des meubles à laquelle appartiennent les logiciels" (Jean-Paul Smets-Solanes, Benoît Faucon, 1999, p. 75).

Enfin, par rapport au succès relatif des logiciels libres (cf. chapitre VIII) et à la tendance à s'appuyer sur les brevets plutôt que sur les droits d'auteur pour garantir les droits de

[51] GNU est un acronyme récursif signifiant GNU's Not Unix.

[52] Richard Stallman parle de *copyleft* !

[53] Toutefois à côté de la licence GPL, il existe une licence plus souple (Library GPL) destinée aux bibliothèques de composants logiciels : celle-ci autorise l'utilisation de composants logiciels libres dans des logiciels "privés" sans être obligé de diffuser le code source pour l'ensemble du logiciel intégrant certains de ces composants. La création de cette licence, qui apparaît contradictoire avec l'éthique du logiciel libre, se justifie pour des raisons stratégiques, à savoir faciliter l'utilisation et la diffusion des composants logiciels libres (Richard Stallman, 1999).

[54] Richard Stallman (1999) cite comme exemple de privatisation d'un logiciel du domaine public, l'interface Unix, X-Windows, initialement distribuée par le MIT avec une licence très permissive et qui a été

propriété, se sont développées récemment de nouvelles formes juridiques basées sur des licences spécifiques (*Netscape Public Licence, Sun Community Source License, Jini Technology Public Licence*…) combinant publicité du code source des logiciels, versement de redevances en cas d'utilisation commerciale, et procédures de certification. Ces licences s'accompagnent de dépôts de brevets effectués par ces sociétés[55].

En conclusion, le caractère de bien partiellement collectif des progiciels a débouché sur des statuts juridiques variés et non stabilisés, les catégories juridiques habituelles ayant du mal à s'adapter aux spécificités des logiciels. Nous allons rencontrer d'autres difficultés pour appliquer les théories économiques destinées à expliquer la valeur et les prix des produits.

C - LES DIFFICULTES POUR APPREHENDER LA VALEUR DES LOGICIELS

La détermination des prix des logiciels apparaît à bien des égards singulière : très grande variabilité des prix dans le temps et selon les utilisateurs pour les progiciels, domination de la "régie" et difficultés à instaurer une tarification au "forfait" pour les logiciels sur mesure (1). Pour les logiciels, les différentes théories de la valeur élaborées pour expliquer les prix des marchandises, qu'elles soient fondées sur la valeur d'usage du produit (2) ou sur la quantité de travail (3) apparaissent inadaptées.

1 - Les singularités de la détermination des prix des logiciels

Une particularité des logiciels est que le système des prix apparaît comme un argument relativement secondaire de la prise de décision (importance de la compétitivité hors prix) , comme un vecteur d'information assez pauvre, à l'encontre du postulat hayekien, que ce soit dans le cas des progiciels (extrême variabilité) ou des logiciels sur mesure (tarification en "régie").

a - L'extrême variabilité des prix des progiciels

appropriée par des promoteurs des environnements Unix "propriétaires" : ceux-ci ont intégré cette interface à leur système sous forme de code binaire exécutable sans diffuser le code-source.

[55] Ces formes de logiciels "pseudo-libres" ont suscité la création de l'*Open Source Initiative*, présidée par Eric S. Raymond, dont l'objectif est de défendre le label "logiciel libre" contre les tentatives de récupération (Jean-Paul Smets-Solanes, Benoît Faucon, 1999, p. 127).

Celle-ci s'explique fondamentalement par la nature de bien intangible, et les caractéristiques de biens collectifs qui en découlent, des progiciels. Sans reprendre l'analyse effectuée précédemment, il faut souligner l'importance des coûts irrécupérables (l'essentiel des dépenses concerne le développement de l'original, sont donc engagées avant de vendre le premier exemplaire et ne peuvent être récupérées en cas de renoncement au projet, comme pour les biens matériels où il est en général possible de les revendre) et les difficultés d'appropriation. Ceci explique qu'il soit fréquent d'inclure la vente du progiciel soit dans la vente du matériel[56], soit dans la vente d'un autre progiciel plus indispensable[57], soit (plus récemment) dans la vente de services, sans que le prix du progiciel considéré apparaisse clairement pour l'acheteur. Il faut également remarquer qu'avec le développement de la vente de progiciels par Internet, le prix réel payé par l'utilisateur n'est connu qu'après coup, dépend des compétences et de l'équipement de l'acheteur, et est une combinaison, souvent indistincte, de redevances, de charges d'utilisation et de coûts de communication (Herbert R. Brinberg, Thomas E. Pinelli et Rebecca O. Barclay, 1997, p. 184).

L'importance des coûts fixes, indépendants des quantités vendues, explique que le prix d'un produit, élevé lors de son lancement pourra baisser fortement avec l'élargissement de sa diffusion, l'élasticité-prix de la demande étant élevée (OCDE, 1997 B, p. 22) : par exemple, Eurostaf estime que le prix d'un *firewall* (logiciel de protection d'un réseau interne connecté à l'extérieur) devrait baisser de 16 000 dollars à 650 dollars entre 1996 et 2000, ses ventes prévues passant de 40 000 unités à 1510 000 sur la même période (1997 A, p. 25). Il faut également prendre en compte les stratégies des producteurs liées à la force des rendements croissants d'adoption dans la diffusion des progiciels. Pour s'imposer sur un marché, une entreprise peut commercialiser un progiciel à un prix extrêmement bas, voire le distribuer gratuitement (stratégie de Microsoft pour imposer son navigateur sur un marché dominé à l'époque par Netscape), en espérant compenser le manque à gagner par une augmentation des prix quand il se sera retrouvé en position hégémonique[58]. La spécificité de ces marchés, de

[56] Il est par exemple extrêmement difficile d'acheter un micro-ordinateur sans un certain nombre de progiciels, même si l'acheteur les possède déjà ou ne souhaite pas les acquérir.

[57] Le cas le plus célèbre et le plus contesté est l'intégration par Microsoft de son navigateur (Internet Explorer) dans son système d'exploitation (Windows 98).

[58] Par contre, une politique de prix très bas une fois que les anticipations ont convergé sur un produit concurrent sont rarement efficaces, une trop forte baisse de prix pouvant même être interprétée comme le signe que l'entreprise est vaincue. Carl Shapiro et Hal R. Varian citent les exemples de Computer Associates avec le

type "*Winner Take All*", peut susciter le développement de pratiques surprenantes et parfois contestables[59] : par exemple Microsoft facturait la vente de son système d'exploitation, de 1990 à 1994, aux constructeurs de micro-ordinateurs en fonction du nombre total de micro-ordinateurs vendus, indépendamment du fait que le micro-ordinateur vendu comprenne le système d'exploitation de Microsoft, un système d'exploitation produit par une autre société ou aucun système d'exploitation[60]. De même, la rapidité avec laquelle apparaissent les nouveaux produits et les nouvelles versions, et l'insignifiance du coût marginal d'un nouvel exemplaire, peuvent conduire les entreprises à "brader" littéralement certains produits : il n'est pas rare de voir certains produits, vendus plusieurs milliers de francs, être proposés par certains canaux de distribution quelques dizaines de francs quelques années après leur lancement[61].

La deuxième dimension de la variabilité des prix des progiciels réside dans la pratique d'une tarification différenciée selon le type d'utilisateurs[62]. Celle-ci peut prendre des formes multiples : rabais consentis aux possesseurs d'une ancienne version du même produit ou d'un

logiciel Simply Money distribué gratuitement (à l'exception des frais de transport) qui n'a pas soutenu la concurrence de Quicken et de Money, d'OS/2 bradé à 50 dollars par IBM sans succès face à Microsoft, ou du tableur de Borland face à Lotus et Microsoft (1999, p. 252).

[59] "Le coût élevé de production du premier exemplaire d'un bien ou d'une technologie d'information engendre inévitablement des stratégies de différenciation du prix et du produit. Les stratégies de personnalisation à grande échelle, de discrimination par le prix, de personnalisation du contenu et du produit sont le résultat naturel de cette structure de marché. Ces stratégies peuvent cependant mettre la concurrence en question de façon sensible" (Carl Shapiro, Hal R. Varian, 1999, p. 263).

[60] Sommé par la justice de renoncer à imposer ce type de pratique aux constructeurs, Microsoft a adopté une double tarification qui apporte peu de changements dans les faits : un tarif peu élevé pour les constructeurs qui continuent à pratiquer l'ancien système (nombre de machines vendues) et un tarif beaucoup plus élevé pour les constructeurs qui ne paieraient que les systèmes d'exploitation Microsoft fournis aux utilisateurs. Dans une situation où la concurrence sur les prix est devenue particulièrement rude entre les constructeurs et où les marges se sont érodées, cette dernière possibilité n'a pas été adoptée par beaucoup de constructeurs. Microsoft ne semble pas vouloir abandonner ce type de pratiques, comme l'illustre les récentes propositions aux établissements scolaires d'une tarification ("licence School pour l'éducation") basée sur le nombre d'ordinateurs de chaque établissement scolaire ("ordinateurs éligibles") indépendamment des logiciels installés sur ces ordinateurs, et aux universités ("licence Campus") d'un prix basé sur le nombre d'enseignants et d'étudiants.

[61] En France, une revue au titre significatif ("Presq'offert") s'est spécialisée dans ce type de ventes.

[62] Aux Etats-Unis, la discrimination par les prix est illégale selon le *Robinson-Pattman Act* de 1936 lorsqu'elle entraîne "une réduction effective de la concurrence" ; toutefois il est possible de fixer des prix différents si les coûts de production sont différents, pour faire face à la concurrence. Ce n'est que si cette pratique aboutit à réduire la concurrence qu'elle peut engendrer des sanctions juridiques (Carl Shapiro, Hal R. Varian, 1999, p. 264).

produit concurrent, prix beaucoup plus faibles lors d'achat en grande quantité[63], prix différents selon le type de matériel auquel est destiné le progiciel[64] ou selon le type d'utilisation présumée, fourniture de prestations complémentaires "gratuites" d'importance différente selon les clients[65]. Cette tarification par segments est permise par le fait que pour les progiciels "le contrat de licence accorde à l'utilisateur un droit d'usage définitif, sans lui en transférer la propriété dans la mesure où il ne peut le revendre à un tiers" (Gérard Dréan, 1996 A, p. 189). De ce fait, il devient possible pour le producteur de pratiquer une discrimination par les prix (Godefroy Dang Nguyen, 1995, p. 215-216), qui en ajustant plus ou moins finement les prix à la propension à payer de chaque segment de clientèle, permet de récupérer une partie du surplus des consommateurs (la différence entre le prix qu'étaient prêts à payer certains consommateurs et le prix unique du produit dans le cas d'une impossibilité de discrimination). Une autre possibilité pour arriver au même résultat consisterait à différencier le produit en l'adaptant à l'utilisation distincte des consommateurs (Jean-Pierre Angelier, 1997, p. 118), ce qu'avait fait par exemple Microsoft en développant une version aux fonctionnalités réduites de son traitement de texte (Word Junior). Elle présente toutefois l'inconvénient de développer et de gérer simultanément plusieurs produits différents et donc de ne pas bénéficier pleinement des économies d'échelle possibles[66]. Dans le cas des progiciels, il est possible et souvent plus

---

[63] Le même produit (la suite bureautique Office de Microsoft) peut se voir proposé à un prix variant de 50 F. pour les "grands comptes" à 4 000 F. pour une P.M.E. (Jean-Paul Smets-Solanes, Benoît Faucon, 1999, p. 112).

[64] De façon générale, un progiciel destiné à être utilisé sur une station de travail ou sur un miniordinateur est vendu beaucoup plus cher qu'une version identique pour un micro-ordinateur.

[65] Cette pratique était l'œuvre d'IBM dans les années soixante pour les ordinateurs lorsqu'elle ne facturait pas les développements logiciels. Elle disposait ainsi "d'un moyen sans faille de modulation de ses prestations "à la tête du client", car il était difficile de mesurer ce que lui coûtait réellement ce service, qu'elle accordait de façon préférentielle à ses meilleurs clients" (Godefroy Dang Nguyen, 1995, p. 329).

[66] Dans le domaine du matériel la possibilité de concilier économie d'échelle et discrimination des prix est parfois réalisée par des artifices techniques : par exemple, pour certaines gammes d'ordinateurs IBM ne produisait et ne livrait que le plus puissant qu'elle "bridait" éventuellement (pratique appelée du "tournevis d'or"). De façon plus générale, il était fréquent qu'une machine moins puissante, et donc offerte à un prix inférieur, soit obtenue en ajoutant à la machine initiale un dispositif de ralentissement ou autre limitation, ce qui augmente donc son prix de revient par rapport à la machine initiale ; par exemple, l'imprimante LaserPrinter série E d'IBM est fonctionnellement identique à la LaserPrinter standard vendue plus cher, mais comprend en plus une puce programmée pour diviser par deux le temps d'impression (Carl Shapiro, Hal R. Varian, 1999, p. 58). Gérard Dréan parle de tarification "fonctionnelle" pour désigner cette pratique, qui "permet au constructeur d'empocher la totalité de la différence entre la valeur d'usage que présente le produit pour son utilisateur et le coût de revient de ce produit" (1996, p. 18).

rentable de vendre le même produit à des prix différents selon des profils d'utilisateurs[67], ce qui constitue une des explications du fait que fréquemment une partie minime des fonctionnalités d'un progiciel est effectivement utilisée.

L'ensemble de ces pratiques explique que les prix des progiciels "sont à peu près indépendants du contenu fonctionnel du produit et donc de ses coûts" (Gérard Dréan, 1996 A, p. 223). Enfin, il faut noter que se développe la location des logiciels en ligne effectuée par les *Application Service Providers* (ASP), par exemple la société Hot Office Technologies qui loue un logiciel semblable à Lotus Note (Les News, 13/09/99). Des acteurs puissants (Sun, Microsoft) ont annoncé leur intention de recourir à ce mode de commercialisation. Le succès de ce modèle économique, où le logiciel ne fonctionne plus sur la machine de l'utilisateur mais sur un serveur Web, est toutefois subordonné à l'avenir de la conception de l'informatique dite *"Network Centric Computing*" qui n'est nullement assuré (cf. chapitre II).

b - Les pratiques dominantes de la "régie" et les difficultés du "forfait" pour les logiciels sur mesure

La production de logiciels sur mesure est fondamentalement une activité de service. Pour les logiciels développés en interne, l'existence éventuelle d'un prix est de nature essentiellement conventionnelle. Dans les cas où ces logiciels sont développés par un prestataire extérieur, il existe deux grands modes de facturations : la pratique de la "régie" qui consiste pour le prestataire à facturer les moyens essentiellement humains mis à la disposition du client qui demeure maître d'œuvre de la prestation (le prestataire a seulement une obligation de moyens), et la pratique du "forfait" où le prix d'une prestation précisément définie est fixé préalablement à l'exécution de celle-ci (obligation de résultats : le prestataire doit réaliser un projet donné en respectant les coûts et les délais établis avec le client, voire même des engagements de fonctionnement et de performances). En théorie, le forfait (connaissance des caractéristiques du produit et de son prix *ex ante*) est une procédure qui

[67] Toutefois il existe parfois également des versions "dégradées" de logiciels où certaines fonctions ont été désactivées (ce qui signifie par ailleurs qu'elles sont plus coûteuses). Carl Shapiro et Hal R. Varian citent les exemples du logiciel Mathematica, pour lequel il existe une version "étudiant" où certaines fonctions ont été désactivées pour ralentir l'exécution et la visualisation des graphiques, le logiciel PhotoDeluxe version "allégée" de Photoshop d'Adobe, et la version Windows NT Workstation qui est la même que la version Serveur, beaucoup plus chère, mais qui est "bridée" ce que confirme, malgré les dénégations de Microsoft, le fait que certains administrateurs de réseaux aient réussi à la "débrider" (1999, p. 63 - 69).

correspond au rôle attribué au mécanisme des prix dans une économie concurrentielle. Pourtant dans la pratique, c'est le mode de la régie qui est largement dominant.

En effet, le développement d'un logiciel sur mesure est "un éventail étendu d'activités qui part de la naissance d'un besoin d'un utilisateur, très flou, immatériel et instable, pour se terminer par l'exploitation d'un « produit » à caractère industriel qui doit être performant, documenté et robuste à l'exploitation". Entre ces deux extrêmes, "tout un cheminement intellectuel et industriel, la démarche d'ingénierie, permet de progresser par itérations vers des représentations de l'objet « produit-service » de plus en plus matérielles, adaptées et stables." (Serge Bouchy, 1994, p. 174). Il est quasiment impossible même en utilisant des méthodes très formalisées et en développant des cahiers des charges volumineux d'établir préalablement à la prestation des spécificités exhaustives et intangibles de celle-ci. L'établissement progressif de ces spécificités représente une part de plus en plus importante de l'activité, et les méthodes itératives plus récentes visant à mieux répondre aux besoins des utilisateurs intègrent une évolution inévitable des caractéristiques de l'application dans un processus d'interaction permanente avec les utilisateurs. Le développement de logiciel sur mesure est typiquement une activité complexe correspondant à un mandat confus (Jacques Girin, 1994, p. 19), où un contrat est nécessairement incomplet, les co-contractants n'étant pas en mesure de préciser avec exactitude, le résultat attendu, l'objet sur lequel portera l'évaluation du résultat et les moyens de vérifier les engagements, d'autant plus que le mandant participe à la prestation et a donc une part de responsabilité dans la réalisation de celle-ci.

De ce fait, il est assez inévitable que les contrats soient centrés sur la définition des moyens mis à disposition (modèle de la régie), rares étant les entreprises qui travaillent réellement sur la base de contrats de résultats forfaitaires (Gérard Dréan, 1996 A, p. 279). L'existence de prestations au forfait est souvent de la "régie déguisée" avec une succession de forfaits qui évoluent au fur et à mesure du développement de la prestation. C'est notamment le cas pour les projets de taille importante où certes existent des études de faisabilité et des études préalables avant la signature du contrat, mais qui concernent justement les types de projets qui voient les dérives les plus importantes par rapport aux spécifications initiales[68].

[68] Un exemple des difficultés de tels contrats est la rupture spectaculaire survenue en pleine réalisation d'un projet entre la Bibliothèque Nationale de France et Cap Gemini sur un contrat d'un montant initial de 138 millions de francs. La BNF reproche le manque de fiabilité et les retards de livraison des premières versions

Pour l'analyse des contrats en matière de développement de logiciel sur mesure, il faut prendre en compte l'importance de l'asymétrie informationnelle et les problèmes d'incertitude (S. Wang, 1992), les phénomènes d'*aléa moral* et de *sélection adverse*, prenant une dimension majeure. Dès lors le prix ne peut être un indicateur suffisant pour orienter le comportement des clients dans la concurrence[69]. Pour que les prix déterminent l'attitude des consommateurs, il est nécessaire qu'il y ait homogénéité du produit. Or, même lorsqu'il existe plusieurs propositions substituables au sens où elles visent à répondre aux mêmes besoins, elles ne sont pas homogènes car elles ne présentent pas en général les mêmes caractéristiques. La production des logiciels sur mesure combine les caractéristiques des différents cas où, selon Jacques Girin, la question de la connaissance de la compétence du fournisseur ne peut être éludée : mandat nouveau à une entité sans possibilité d'attendre les résultats pour s'assurer des compétences du mandataire, caractère déraisonnable du fait d'attendre les résultats pour évaluer la capacité d'un système à les atteindre, conception des traits généraux du "*design*" d'un sous-ensemble en vue d'une tâche déterminée, question de la performance qui ne se pose pas de manière simple et qui ne peut se résumer facilement dans quelques indicateurs physiques ou financiers (1994, p. 26). La compétence du prestataire peut être particulièrement difficile à évaluer directement par le client et des phénomènes de nature conventionnelle (réputation, confiance, certification) vont modeler fortement les comportements dans le secteur.

2 - Les difficultés pour appréhender la valeur d'usage d'un logiciel

La notion de valeur d'usage est au centre de l'explication des prix par la théorie néoclassique de base avec les notions de coût marginal croissant et de rareté. La question de la rareté se pose de façon particulière pour les progiciels, comme de façon plus générale pour les biens intangibles existant sous forme numérique (Blaise Cronin, 1997, p. 15-16), dans la mesure où la rareté ne peut être créée qu'artificiellement, en conséquence d'un coût marginal qui tend vers zéro avec les quantités produites. "La production d'un bien d'information nécessite des coûts fixes élevés, mais a un coût marginal faible", ce qui implique "qu'il est

(jusqu'à 22 mois), Cap Gemini mettant en avant les modifications du cahier des charges (accroissement des fonctionnalités qui impliquerait une surcharge de travail de 50 %) (01 Informatique, 30/07/1999).

[69] Pour les services de façon plus générale, Jacques De Bandt et Jean Gadrey s'interrogent : "dans quelles conditions peut-on encore parler de marché (compte tenu de l'incertitude sur l'objet et l'enjeu de la transaction),

impossible de fixer le prix d'un tel produit en se basant sur le coût de production" et "qu'il faut fixer le prix en fonction de la valeur du bien pour les consommateurs" (Carl Shapiro, Hal R. Varian, 1999, p. 9-10)[70]. La question que nous voudrions aborder maintenant est celle de la difficulté qu'il y a à appréhender la valeur d'usage d'un logiciel[71], indépendamment du caractère pertinent de ce cadre théorique dans l'explication de la détermination des prix pour l'ensemble des biens et services.

a - Les difficultés à isoler l'impact d'un composant d'un bien-système mobilisant les compétences des utilisateurs

La première difficulté est qu'un logiciel est nécessairement utilisé de façon indissociable d'autres produits logiciels et matériels qui déterminent les caractéristiques de son environnement (cf. chapitre I). Cette situation est poussée à l'extrême dans le cas des "systèmes propriétaires" où un logiciel ne peut fonctionner que dans un type d'environnement matériel et logiciel limité. L'ouverture des systèmes a atténué ces dépendances, mais il n'en demeure pas moins que ce que peut appréhender l'utilisateur c'est l'efficacité globale d'un système constitué de composants matériels, logiciels et de services. L'évolution des logiciels, qui de plus en plus réalisent des fonctions complexes, renforce cette tendance. Ce qui devient déterminant c'est plus l'efficience de l'interopération entre les différents composants que l'efficience de chaque composant pris isolément. Par exemple, un échec dans une recherche d'informations sur le Web peut résulter d'une défaillance de la coordination des différents éléments concernés très divers[72] ou d'un des éléments du système, sans possibilité pour l'utilisateur d'identifier le produit incriminé.

---

de prix (vu la difficulté à définir des unités de produits), de concurrence (lorsque les performances sont évaluées de façon anticipée) ?" (1994, p. 16)

[70] C'est selon ces auteurs ce qui explique que "le prix sera nécessairement fixé de façon discriminante", puisque "les individus valorisent très différemment les biens d'information" (Carl Shapiro, Hal R. Varian, 1999, p. 10).

[71] Par certains aspects, le logiciel est confronté à des difficultés identiques que le produit de la recherche-développement qu'Olivier Weinstein a analysé en soulignant "l'indétermination de sa valeur d'usage" (1989, p. 82).

[72] Sans prétention à l'exhaustivité, on peut mentionner le matériel et les logiciels (système d'exploitation, outils de navigation et de communication) du "client", du "serveur", et les différents composants matériels et logiciels qui constituent l'infrastructure du réseau.

De plus la réalisation des performances potentielles du système dépend de manière importante des compétences de l'utilisateur et de la qualité des services qui contribuent à développer ces compétences. Selon Paricia D. Fletcher et Lester Diamond (1997, p. 176), "la véritable ressource réside dans les hommes qui sont capables d'exploiter les technologies de l'information". A la notion de valeur d'usage incorporée dans le produit lors de sa création, d'une utilité objectivée intrinsèque au produit, Orio Giarini et Walter R. Stahel (1990) proposent de substituer la notion de valeur d'utilisation d'un système, qui est l'utilité obtenue d'un système pendant l'ensemble de sa durée de vie, intégrant le rôle actif du consommateur, caractérisé comme étant un *prosumer* (consommateur-producteur) par Alvin Toffler. Cette valeur dépend des performances d'un agencement organisationnel particulier, composite constitué d'éléments hétérogènes (matériels, logiciels, humains) reliés entre eux (Jacques Girin, 1994, p. 25). Elle est difficilement mesurable objectivement et ne peut le plus souvent qu'être évaluée par des "mesures d'ancrage" ou indicateurs opérationnels de performance qui dépendent des objectifs de l'organisation (Herbert R. Brinberg, Thomas E. Pinelli et Rebecca O. Barclay, 1997, p. 188).

b - L'insuffisance de l'appréciation par les caractéristiques d'usage directes et la nécessité de recourir aux caractéristiques d'usage indirectes

De ce point de vue les indicateurs pertinents pour l'organisation concernent moins les caractéristiques d'usage directes (ou efficacité) que les caractéristiques d'usage indirectes (ou effectivité).

Les caractéristiques d'usage directes sont tout d'abord difficiles à mesurer : à la différence des composants matériels où l'on peut effectuer des mesures physiques des différentes dimensions de l'efficacité, les techniques de mesure (les "métriques") des facteurs de qualité, exemptes autant que faire se peut de subjectivité, ne progressent que très lentement pour les logiciels (Serge Bouchy, 1994, p. 193). Ensuite, ces caractéristiques évoluent dans le temps en fonction de plusieurs facteurs non indépendants : les logiciels sont des "produits évolutifs, pour lesquels l'échange perdure au-delà de la première acquisition pour intégrer les possibilités de leur évolution" (Anne Mayère, 1997 A, p. 131), les environnements matériels et logiciels se modifient, les compétences des utilisateurs changent. Par exemple, Francis Pavé (1989) à partir d'une étude sociologique d'une dizaine de projets informatiques constate que les services utilisateurs rejettent, neutralisent ou dévoient le système, et conclut que les projets réussis sont les projets qui ont été dévoyés car ils ont été réappropriés par les agents. D'autres

facteurs interviennent de façon plus indirecte : l'utilité d'un logiciel pour son détenteur peut être modifiée par son adoption par d'autres acteurs, de façon positive (rendements croissant d'adoption et externalités de réseaux) ou négative quand sa possession exclusive était un facteur de compétitivité par rapport aux entreprises concurrentes (exemple des systèmes de réservation aérienne). En reprenant les distinctions établies par P. Nelson (1970), on peut dire que les logiciels s'apparentent davantage aux "*experience goods*" dont les qualités ne se révèlent qu'au cours de leur utilisation, voire aux "*credence goods*" dont les qualités sont difficilement évaluables par l'utilisateur, qu'aux "*search goods*". Dans le cas plus général des biens d'information, qui sont des biens d'expérience, "les consommateurs se fondent souvent sur la *marque* et la *réputation* du produit" (Carl Shapiro, Hal R. Varian, 1999, p. 11).

Mais, outre leurs difficultés d'évaluation, les caractéristiques d'usage directes ne sont pas l'unique déterminant des caractéristiques d'usage indirectes qui sont *in fine* celles qui sont décisives pour l'utilisateur (le logiciel se révélant fondamentalement plus ou moins utile selon son impact, la réponse qu'il apporte aux objectifs de son acquisition). Interviennent également et de façon encore plus importante que pour les caractéristiques d'usage directes, les caractéristiques de l'environnement non exclusivement informatique et les compétences des utilisateurs pour exploiter les potentialités du logiciel. Par exemple, et même si le manque de compétences des utilisateurs n'est pas la seule cause, Eurostaf estime que les utilisateurs ne se servent en moyenne que de 20 % des fonctionnalités d'un progiciel (1996 C, p.79). Les trois dimensions principales des caractéristiques d'usage indirectes sont l'augmentation de la productivité consécutive à l'introduction du logiciel (ou productivité indirecte, Jacques De Bandt, 1995, p. 162), l'amélioration de la qualité des actions des utilisateurs, le raccourcissement des délais de production et de livraison des utilisateurs[73]. Ce qui va rendre particulièrement problématique la connaissance de la valeur d'usage d'un logiciel par l'utilisateur, c'est que les caractéristiques d'usage indirectes sont encore plus délicates à évaluer, ce qu'illustre les discussions sur le paradoxe de Solow à propos des effets de l'informatique.

c - Les effets de l'informatique et le "paradoxe de Solow"

[73] En se limitant aux cas, de loin les plus fréquents, où l'action peut être effectuée de façon non informatisée. Dans les situations où cette possibilité n'existe pas (par exemple, poursuite d'une cible à partir d'entrées radar, analyse de clés cryptographiques, tomographie médicale…) l'évaluation de l'apport de l'utilisation d'un logiciel devient singulièrement délicate.

L'analyse s'est tout d'abord focalisée sur les caractéristiques d'usage concernant la productivité indirecte : "l'enjeu du logiciel est justement la rationalisation qu'il permet de l'activité économique et notamment du tissu industriel. Concevoir un logiciel, c'est inventer un procédé destiné à améliorer la productivité d'une fonction donnée" (Christine Babelon, 1987, p. 7). Mais le fait qu'une part importante des logiciels (logiciels systèmes, outils de développement, bases de données communes aux applications, bibliothèques de composants…) n'ont de valeur que par rapport aux logiciels applicatifs qu'ils permettent de produire et de faire fonctionner, et la nécessité de prendre en compte également l'infrastructure matérielle nécessaire (cf. a), font que les analyses ont porté essentiellement sur l'impact en termes de productivité de l'informatique prise globalement.

Dans les années soixante-dix et quatre-vingt, la plupart des études prévoyaient que l'utilisation de l'informatique allait entraîner une forte augmentation de la productivité, notamment dans le travail de bureau, un travail que l'ordinateur devrait transformer comme la machine à vapeur et le moteur électrique avaient modifié le travail en atelier (Jean-Louis Peaucelle, 1998, p. 1). Les différentes études divergeaient sur les conséquences positives (augmentation du temps libre) ou négatives (crainte du chômage) de cette évolution de la productivité du travail, mais sa croissance semblait inéluctable. Ces gains de productivité devaient résulter de la réutilisation des données dans un grand nombre d'usages, des possibilités de leur communication à un grand nombre d'utilisateurs, de la capacité à retrouver l'information pertinente dans une grande masse de données, de l'automatisation des calculs et d'une réorganisation des processus de travail ou *Business Process Reengineering* (Jean-Louis Peaucelle, 1998, p. 2).

Toutefois, au fur et à mesure de la diffusion massive de l'informatique, des doutes vont s'exprimer sur la réalité de cette relation, doutes que résume le "paradoxe de Solow" : "*on voit des ordinateurs partout sauf dans les statistiques de productivité*". Ces doutes s'appuient sur un certain nombre d'études empiriques qui mettent en évidence une absence de concomitance entre dépense informatique et productivité à différents niveaux. Sur un plan sectoriel, on peut citer aux Etats-Unis les études de R.H. Franke (1987) pour les banques et les assurances, de L. Thurow (1987) sur les services, de S. S. Roach (1989) sur les cols blancs. Pour la France, une évaluation de seize applications dans les principaux ministères, effectuée à la demande du Comité Interministériel de l'Evaluation des Politiques Publiques (Commissariat Général du Plan, 1992) souligne la faiblesse des gains de productivité réalisés. Le constat qu'entre 1982 et

1990 ce sont les emplois de secrétaires qui ont le plus augmenté en France en volume (+ 256 000, soit + 35 %, source INSEE) malgré l'introduction massive de la bureautique va dans le même sens. Les observations macroéconomiques effectuées dans différents pays montrent également une absence de lien entre une augmentation faible et erratique de la productivité apparente du travail et une croissance forte et continue de la dépense informatique en valeur, sa croissance en volume étant encore plus rapide en raison de l'importance de la baisse des prix (cf. pour la France, sur la période 1986-1995, Jean-Louis Peaucelle, 1997, p. 16). Pour l'ensemble des pays de l'OCDE, le PIB par tête qui croissait au rythme de 3,8 % l'an entre 1960 et 1973, n'augmentait plus que de 1,4 % l'an entre 1979 et 1993 ; de plus ce phénomène était plus marqué pour les services (de 2,7 % à 0,8 %) que pour l'industrie (de 5 % à 2,9 %) alors que ce sont les services qui ont connu l'informatisation la plus importante (Godefroy Dang Nguyen, Pascal Petit, Denis Phan, 1997, p. 65).

Au niveau microéconomique, Jean-Louis Peaucelle à partir de l'examen des différentes études menées qui présentent des résultats contradictoires, d'amplitude limitée et difficilement interprétables, conclut que "l'impact économique de l'informatique doit être faible et, avec nos méthodes actuelles, on ne le distingue pas nettement" (1997, p. 18). Ce paradoxe n'est pas seulement un phénomène statistique mais recouvre également les interrogations des utilisateurs, organisations ou individus, sur l'efficacité réelle apportée par les technologies de l'information et de la communication (Alain Rallet, 1997, p. 85).

Les explications du paradoxe de Solow

Les explications de ce paradoxe apparent ont suscité de nombreux débats. Une interrogation préalable porte la réalité statistique du paradoxe en émettant des sérieux doutes sur la pertinence des statistiques de productivité, notamment dans les activités de services (Jean Gadrey, 1996 A), où "les mesures existantes de productivité [sont] inadaptées pour saisir la forte complexification" des activités (idem, p. 227) ou donne des résultats contradictoires[74]. De même, Pascal Petit estime que "après deux décennies de débat sur le ralentissement des gains de productivité, la qualité de la mesure des gains de productivité reste

[74] Par exemple pour l'activité bancaire, pendant une période de forte informatisation, la productivité en valeur (mesurée à partir du produit bancaire) reste relativement stable semblant confirmer un impact limité de l'informatisation, alors que la productivité en nature (mesurée à partir du nombre de comptes) augmente très fortement (Jean Gadrey, 1996 A, p. 111).

une des explications principales mises en avant" (1998, p. 347). Au-delà, plusieurs explications, qui ne sont pas nécessairement contradictoires, ont été proposées.

Une première série d'explications intègre l'existence de gains de productivité qui semblent considérables pour des activités isolées et bien identifiées. Parmi de multiples exemples, on peut citer l'informatisation de la demande des cartes d'identité, passeports et cartes grises de la préfecture de Seine et Marne qui a fait passer le nombre de dossiers traités par jour par chaque employé de 25 à 150 (Le Monde Informatique, 8 avril 1994). De façon plus générale, Jean Gadrey souligne que "de très importants gains de productivité ont été (et seront encore) réalisés, dans des conditions qui peuvent parfois donner lieu à des mesures précises, dans le vaste ensemble des tâches consacrées au traitement d'informations codées, pour un volume donné d'informations de complexité donnée, à traiter sur un mode standardisé : transactions et dossiers standards, production de comptes et de ratios, opérations statistiques, mailings, facturations, etc." (1996 A, p. 213). De multiples raisons peuvent expliquer que ces gains partiels réels ne soient pas plus répandus et semblent s'évanouir quand la productivité est appréhendée à un niveau plus global : le fait que souvent la solution informatique s'ajoute à la solution non informatique au lieu de la remplacer[75], l'élasticité des tâches (Jean-Louis Peaucelle, 1998, p. 6) qui fait que des tâches obligatoires mais non effectuées auparavant faute de temps peuvent être réalisées, la nécessité de nouveaux emplois et de nouvelles tâches liées à l'informatisation[76], l'amélioration de la productivité du travail pour des postes qui ne constituaient pas des goulets d'étranglement (Jean-Louis Peaucelle, 1998, p. 7) et donc sans incidence globale en l'absence d'une modification de l'organisation du travail, le fait, lié au précédent, que l'effectif de certains postes ne peut descendre en dessous de l'unité et qu'il est déterminé en fonction des périodes de pointe.

La deuxième série d'explications est de nature plus sociologique et insiste sur la persistance de coûts liés aux limites cognitives des individus et des collectifs et aux comportements stratégiques des acteurs qui peuvent avoir intérêt à limiter l'accès à certaines informations (Eric Brousseau, 1997, p. 49). L'informatisation initiale, réalisée en renforçant la

[75] Thomas K. Landauer cite comme exemple le courrier électronique qui ne se substitue pas au téléphone et au courrier classique (1997, p. 5).

[76] Le Gartner Group estime que les utilisateurs consacrent une heure par jour à des tâches liées à l'outil informatique et non au contenu de leur travail.

centralisation de l'information, dépossédait les utilisateurs de la maîtrise de ressources informationnelles. L'irruption d'une micro-informatique décentralisée a permis une réappropriation d'une partie de l'information, mais en se réalisant de façon quelque peu anarchique, elle a multiplié les saisies et les traitements individuels, source de coûts additionnels (Marie-Christine Monnoyer-Longe, 1997, p. 111). Symétriquement, des possibilités apportées par la mise en réseau d'équipements informatiques pour une communication étendue d'informations détenues centralement, ne sont pas exploitées, par crainte de l'autonomisation des salariés qui pourrait en résulter. Des coûts supplémentaires peuvent également provenir d'une tendance exagérée à privilégier la forme sur le contenu, et sont difficilement contrôlables en raison de l'indétermination des usages possibles de l'information et de la déstabilisation constante des procédures d'évaluation et de contrôle par l'innovation technologique (Alain Rallet, 1997, p. 95). Il en résulte une "imprévisibilité fondamentale des effets de productivité en raison de la nature intrinsèquement sociale du processus d'informatisation" (François Pichault, 1990, p. 170).

La troisième série d'explication met en cause la mauvaise gestion de l'informatisation (Erik Brynjolfsson, 1993), notamment des erreurs de management qui conduisent à continuer des projets non performants mais dont l'impact est difficile à mesurer pour les gestionnaires. La modestie des gains de productivité serait la conséquence d'un gaspillage de ressources et des dysfonctionnements (Pierre Bonnaure, 1996, p. 67). Dans certaines entreprises et branches de services, il existerait un suréquipement en ordinateurs et en systèmes d'informations et une surcharge informationnelle (Jean Gadrey, 1996 A, p. 151), qui pourrait n'être que transitoire et s'expliquer par la nécessité d'une période d'apprentissage avant d'utiliser valablement les potentialités de technologies nouvelles.

La quatrième série d'explications considère qu'en fait l'objectif d'informatisation d'une organisation n'est pas la recherche de gains de productivité (et donc qu'il n'y aurait pas de paradoxe). Au niveau des administrations, la productivité (mesurée par personne) n'augmenterait que faiblement, car les effectifs seraient plus déterminés par l'histoire et la volonté des supérieurs hiérarchiques de maintenir le nombre de leurs collaborateurs, indication de l'étendue de leur pouvoir, que par la durée des tâches à accomplir (Jean-Louis Peaucelle, 1998, p. 6). Au niveau des entreprises l'informatisation peut être un investissement obligatoire en raison de nécessités administratives (initiatives des pouvoirs publics), des exigences des clients ou des entreprises partenaires (notamment concernant les modalités de

transmission d'informations), et des pressions des entreprises concurrentes[77]. Dans ce cas, l'informatisation améliorerait la compétitivité, en général très provisoirement vu les possibilités d'imitation, pour l'entreprise concernée mais n'aurait pas d'effet au niveau de la branche ou de l'ensemble de l'économie (Alain Rallet, 1997, p. 93), les gains en termes de part de marché s'effectuant dans un jeu à somme nulle sur le plan de la productivité, comme pour la publicité (Eric Brousseau, 1997, p. 62). Beaucoup d'innovations "centrées aujourd'hui sur des logiques de différenciation de produit pouvant aller jusqu'à un degré extrême de superficialité, l'introduction plus rapide des nouveautés, l'anticipation et le contournement des dispositifs réglementaires" présenteraient "des taux élevés de rendements privés" mais ne produiraient que "fort peu d'externalités positives" (Dominique Foray, Bengt Ake Lundvall, 1997, p. 24). Il faut toutefois noter que les études empiriques de Claude Salzman (1982) sur 150 entreprises françaises et de Paul A. Strassmann (1990 et 1997) sur des entreprises américaines ne confirment pas l'influence de l'importance des dépenses informatiques sur la rentabilité des entreprises, qui devrait résulter d'une compétitivité accrue.

La multiplicité d'impacts de l'informatisation

Cependant, cette explication introduit l'idée convaincante et qui s'est imposée progressivement, que l'informatisation a une multiplicité d'impacts, trop longtemps réduits aux seuls gains de productivité (Jean-Louis Peaucelle, 1997, p. 41). Son influence est certainement au moins aussi importante sur les deux autres dimensions des caractéristiques d'usage indirectes que sont l'amélioration de la qualité des actions des utilisateurs et le raccourcissement des délais de production et de livraison. Les changements dans la qualité des biens et services produits, qui peuvent masquer des améliorations réelles de productivité, concernent l'augmentation du standard moyen exigé (par exemple pour la rédaction et la présentation des documents), la variété des produits voire la création de nouveaux produits, qui incorporent de plus en plus de fonctions informatiques. Le raccourcissement des délais peut porter sur l'approvisionnement et sur la fourniture de prestations en continu. Au niveau des processus de production, les effets de l'informatisation ne se limitent pas à l'automatisation totale ou partielle de certaines fonctions, mais concernent également la qualité des décisions prises (procédures de décision, quantité et qualité des informations mobilisées), la rapidité de

[77] Il s'agit alors moins d'évaluer ce que peut apporter l'informatisation que ce que peut coûter la non-informatisation. Ce type d'évaluation est particulièrement délicat.

réaction des entreprises à une évolution rapide de la demande des consommateurs, la fiabilité (notamment à travers les possibilités de traçabilité) et de façon plus générale une flexibilité potentiellement plus importante (Blaise Cronin, 1997, p. 10). Ainsi Jean Gadrey indique qu'il s'est produit "une progression souvent considérable, du volume et de la complexité des informations produites et traitées, en relation étroite avec la croissance de la complexité globale des services offerts : diversification des prestations offertes, complexification de l'environnement réglementaire, segmentation croissante des marchés, adjonction de nouvelles caractéristiques de services, de nouvelles garanties, individualisation des solutions, etc. Les livres de compte ont une progression de 40 % des lignes en dix ans, les dossiers médicaux et administratifs des malades à l'hôpital se sont fortement complexifiés, les stratégies de livraison "juste à temps" ont provoqué un gonflement du volume d'informations qui circulent entre les unités de production et le même phénomène existe dans les entreprises de service organisées sous forme de chaînes et de réseaux" (Jean Gadrey, 1996 A, p. 214). Cette multiplicité d'impacts de l'informatique s'est accentuée avec le passage de l'informatisation initiale de tâches bien identifiées, isolées et délimitées, au développement d'un système intégré d'informations de l'entreprise, de complexité croissante (cf. section I B) et portant sur "les mécanismes de coordination entre les agents, c'est à dire sur la manière dont s'articulent les tâches au sein du processus de travail, les fonctions au sein de la firme, les activités entre les firmes ou entre les secteurs" (Alain Rallet, 1997, p. 95-96).

Le problème est que ces impacts sont difficiles à mesurer directement. Si dans certains cas, on peut disposer d'indicateurs chiffrés significatifs (concernant les délais ou la disponibilité des services), c'est beaucoup moins évident pour évaluer la qualité d'un produit ou d'une décision. Par exemple une étude de l'OCDE sur l'impact des technologies de l'information dans le secteur des biens conditionnés conclut en indiquant que si certains aspects sont mesurables (niveaux des stocks, cas de rupture…) la plupart des changements sont qualitatifs (rapidité, commodité et variété) et défient toute mesure (OCDE, 1997 B, p. 212). De plus les modifications des caractéristiques des biens et des services rendent problématique la mesure de la production "en volume" et donc la perception de l'évolution de la productivité. C'est particulièrement net quand l'analyse est menée à un niveau désagrégé (poste de travail ou processus de production) où la production est mesurée à l'aide d'indicateurs physiques (unités d'œuvre). Quand l'analyse s'effectue à un niveau plus agrégé (entreprise, secteur, pays) ce phénomène est plus limité sans toutefois disparaître (cf. la sous-estimation de l'amélioration de la qualité des produits qui pourrait représenter jusqu'à un

demi-point de croissance annuelle pour les Etats-Unis et qui aurait surévalué de plus de 1 % par an sur deux décennies l'indice des prix à la consommation, selon le rapport de M. Boskin, 1996), mais dans ce cas il est plus difficile d'isoler, parmi les gains, ceux qui proviennent de l'informatisation (Alain Rallet, 1997, p. 88).

Pour résoudre ces problèmes, on peut tenter d'apprécier indirectement les impacts de l'informatisation en examinant ses conséquences sur les performances des entreprises à partir de critères (rentabilité, compétitivité) pour lesquels on dispose d'indicateurs plus fiables. Il faut toutefois toujours mesurer l'informatisation de l'entreprise (l'autre terme de la relation). Or la mesure d'un stock de produits matériels et immatériels, opération toujours délicate, est rendue plus difficile par l'extension et l'intégration de l'informatisation qui a tendance à estomper les frontières entre technologies de l'information d'une part, automatisation industrielle et réseaux de télécommunications d'autre part (Alain Rallet, 1997, p. 88). Surtout, et c'est peut-être l'explication des résultats décevants des études empiriques de Claude Salzman et de Paul A. Strassmann citées précédemment, il peut s'opérer une redistribution des gains apportés par l'informatique au détriment de l'entreprise qui s'est informatisée (Erik Brynjolfsson, 1993). Jean-Louis Peaucelle prend l'exemple de la mensualisation des salaires à la fin des années soixante par versement sur un compte bancaire rendu possible par l'informatisation de cette opération, et qui a certes profité aux entreprises, mais également aux salariés, aux banques et à l'Etat (1997, p. 20). L.M. Hitt et E. Brynjolfsson (1996) ont tenté de démontrer, par des méthodes économétriques, que l'informatisation aurait surtout profité aux consommateurs, notamment par la gratuité de nouveaux services et les baisses de prix. Les avantages pour les clients de l'informatisation peuvent être indirects : par exemple, dans le cas de l'entreprise Districast déjà mentionné, ce sont les entreprises clientes qui bénéficient de l'informatisation du distributeur, qui, en limitant le délai de livraison à 24 heures, leur permet de réduire fortement leurs stocks. A contrario pour Districast ceci peut être la source de coûts supplémentaires dus à la nécessité de gérer de plus petites commandes. (Catherine Palierne, 1998, p. 110). De façon plus générale, l'évolution des systèmes d'information vers l'aval et vers l'extérieur de l'entreprise (Marie-Christine Monnoyer-Longe, 1997, p. 120) accentue le partage des tâches entre les firmes à un point tel qu'il devient impossible d'identifier la contribution spécifique de chaque partie (Pascal Petit, 1998). C'est un constat similaire qu'effectue l'OCDE qui indique que "les technologies de l'information se caractérisent avant tout par les gains d'efficacité globale qu'elles permettent de réaliser en reliant différents

acteurs dans des industries très diverses, et non par leur impact sur des entreprises données" (OCDE, 1997 B, p. 213).

La prise en compte du temps

La dernière explication, mais non la moins importante, de ce paradoxe est la prise en compte de la dimension temporelle, les effets de l'informatisation se manifestant avec retard (Erik Brynjolfsson, 1993). Au niveau microéconomique, une étude de H. Kivijärvi et T. Saarinen (1995) sur 200 entreprises finlandaises montre que ce sont les entreprises qui se sont informatisées depuis le plus longtemps, qui en tirent le plus de profit. Au niveau macroéconomique, la comparaison est effectuée avec l'électricité où il fallut attendre quarante ans pour que l'impact de l'électricité sur la croissance apparaisse dans les statistiques (P.A. David, 1991). Selon S. Roach, "la plupart des TI étant relativement récentes, les managers et les travailleurs informationnels en seraient seulement à un stade intermédiaire de la traditionnelle « courbe d'apprentissage » où l'expérience s'acquiert au prix d'erreurs et de certains gaspillages" (cité par Jean Gadrey, 1996 A, p. 151). Les explications de l'important décalage temporel observé sont dans les deux cas les obstacles à une diffusion généralisée de la nouvelle technologie (Chris Freeman, 1988, p. 54-59) et notamment la lenteur du processus d'apprentissage. La durée de ces apprentissages n'est pas très surprenante en raison du caractère ample et systémique des changements introduits (Pascal Petit, 1998, p. 29). Richard E. Walton (1989) distingue trois composantes d'apprentissage : l'apprentissage technologique (capacité d'utilisation de l'outil), l'apprentissage organisationnel (capacité d'adapter l'organisation de l'entreprise aux nouvelles formes organisationnelles) et l'apprentissage d'affaires (capacité d'élaborer et de concevoir de nouveaux produits et services à partir du savoir-faire technologique et organisationnel). Or ces trois composantes ont une temporalité longue : l'apprentissage de la technologie par les utilisateurs ne s'effectue que progressivement dans un processus d'amélioration et de découvertes de nouveaux usages (Nathan Rosenberg, 1983), les changements organisationnels sont lents, coûteux et difficiles à mettre en œuvre (Eric Brousseau, 1997, p. 41) et la diffusion des technologies de l'information dans de nouveaux produits est encore plus longue que dans les procédés de production (Alain Rallet, 1997, p. 92). De plus, le processus d'apprentissage peut être considéré comme "un processus de rétro-alimentation entre ces trois composantes" (Ramon Salvador Valles, Lucas Van Wunnik, Félix Pineda, 1997, p. 139) et c'est seulement quand ce processus dynamique a permis d'atteindre un nouvel état d'équilibre que sont pleinement exploitées les potentialités

de la nouvelle technologie. Or, la rapidité d'apparition de technologies nouvelles, introduisant des ruptures brutales dans les conditions d'utilisation des technologies de l'information, réactive constamment le cycle expérimentation, rationalisation et stabilisation (Alain Rallet, 1997, p. 86), et a, jusqu'à maintenant, empêché ce processus d'apprentissage de se dérouler complètement.

En conclusion et quelles que soient les explications apportées, il apparaît clairement qu'il est particulièrement difficile d'évaluer *ex ante* l'efficacité des technologies de l'information. Comme le note Frantz Rowe (1997, p. 24), "la productivité de l'informatique n'existe pas (…). On ne peut concevoir une performance de la technique au pied de la lettre, mais une performance appréhendée au niveau du n-uple activité-outil-utilisateur-commanditaire-organisation-client". Isoler au sein des technologies de l'information l'apport d'un logiciel particulier est encore plus problématique. Il en résulte que la valeur d'usage du logiciel peut difficilement servir de cadre théorique pour expliquer l'évolution des prix des logiciels.

3 - La théorie de la valeur-travail, une théorie inadaptée au cas des logiciels

La deuxième explication théorique des prix à partir d'une théorie de la valeur est la théorie de la valeur-travail de Karl Marx. Selon Marx, la valeur d'échange d'une marchandise est déterminée par la quantité de travail (mort et vivant) *socialement* nécessaire à sa production. Socialement signifie en moyenne vu le niveau de développement des forces productives de la société dans laquelle circulent les marchandises. En conséquence, le champ d'application de cette théorie se limite aux produits reproductibles et Marx excluait explicitement le cas de ce que l'on peut appeler des biens tangibles non reproductibles (œuvre d'art originale par exemple).

Il nous semble que ce cadre théorique ne peut pas non plus s'appliquer aux logiciels. Concernant les logiciels sur mesure, qui sont un cas particulier de services non standardisés, le produit étant par définition significativement différent pour chaque prestation réalisée, la valeur d'échange ne peut être fondée que sur la quantité de travail nécessaire dans un cas particulier, ce qui est une autre façon d'expliquer la prédominance de la régie sur le forfait. A l'inverse les progiciels se caractérisent par leur très grande reproductibilité. Mais dans le cas des progiciels, comme dans celui plus général des biens intangibles, la quantité de travail pour réaliser un exemplaire supplémentaire (simple opération de copie) est dérisoire par rapport à

la quantité de travail nécessaire pour développer l'original. De ce fait, la théorie de la valeur-travail n'est pas non plus susceptible d'expliquer les mouvements de prix pour ce type de biens. Il nous paraît plus judicieux d'analyser l'évolution des prix par l'existence d'une rente que pourrait s'approprier le titulaire des droits de propriété sur l'original. Cette rente est une forme particulière de rente différentielle, liée à l'avantage qu'un progiciel donné peut détenir par rapport à des produits concurrents, en termes de caractéristiques d'usage, cet avantage pouvant résulter de ses qualités propres mais également de l'importance de sa diffusion, en raison de la force des rendements croissants d'adoption. Ceci peut permettre d'expliquer les stratégies de prix bas, voire de gratuité, pratiquées pour s'imposer sur un marché et se retrouver ultérieurement dans une situation permettant de capter une rente importante. En effet, quand un progiciel a réussi à devenir hégémonique sur un segment de marché, processus qui peut être rapide et qui est difficilement réversible, ce progiciel peut être vendu à un prix élevé (rente différentielle très importante), ce qui se traduit par le très haut niveau de profitabilité que peuvent atteindre certains éditeurs de progiciels. Réciproquement, l'apparition d'un nouveau produit concurrent plus performant qui réussit à conquérir une part significative de marché, peut faire disparaître la rente différentielle dont bénéficiaient les produits précédents, expliquant les chutes brutales de prix constatées, qui peuvent aller jusqu'à la disparition d'entreprises très profitables précédemment (cf. certains éditeurs de progiciels pour micro-ordinateurs comme Aston-Thate pour les bases de données ou WordPerfect pour les traitements de texte). Ce type de rente, que l'on peut assimiler à une rente technologique, existe dans tous les secteurs où les coûts de conception représentent une part importante des coûts (matériel informatique par exemple). Mais dans le cas des biens tangibles, il faut également intégrer les coûts de l'activité de fabrication *stricto sensu* (la réalisation d'un exemplaire supplémentaire) qui sont insignifiants dans le cas des biens intangibles.

On peut également noter que la place prise par les logiciels dans la production a des conséquences essentielles pour la théorie de la valeur-travail concernant l'ensemble des marchandises. L'intuition de cette importance se trouve dans certains écrits de Marx, qui ne concernaient évidemment pas les logiciels, mais le *savoir abstrait*. Dans le "Fragment sur les machines", Marx défend l'idée que le savoir abstrait tend à devenir, en vertu précisément de son autonomie par rapport à la production, la principale force productive, reléguant dans une position marginale le travail parcellisé et répétitif. Pour Marx, il s'agit du savoir objectivé dans le capital fixe, qui s'est incarné dans le système automatique des machines. Marx pour désigner ces connaissances abstraites parle de *general intellect* (cerveau général), ce que

Maximilien Rubel traduit par "puissance matérialisée du savoir". Paolo Virno (1992) propose d'élargir la notion de *general intellect* au-delà de la connaissance qui se matérialise dans le capital fixe, en y incluant aussi les formes de savoir incarnées dans le travail vivant qu'il appelle intellectualité de masse, "qualité et signe distinctif de toute la force de travail de l'époque post-fordiste (…) dans laquelle l'information, la communication jouent un rôle essentiel dans chaque repli du procès de production". Il nous semble plus significatif des transformations récentes de la production d'inclure dans le *general intellect* la part des connaissances codifiées non intégrées directement dans des machines mais qui peuvent circuler et être reproduites indépendamment de la force de travail et qui sont notamment présentes sous la forme de logiciels divers. Cette importance est analysée dans un autre cadre théorique par Herbert Simon (1992) qui parle des logiciels comme facteurs de production. La production de marchandises repose dans une proportion croissante sur des connaissances codifiées qui ne nécessitent quasiment pas de travail pour être reproduites ("librement" reproductibles). De ce fait, la mesure, dans le cadre du système capitaliste, de la valeur par la quantité de travail socialement nécessaire devient de plus en plus contradictoire avec une production qui s'appuie de plus en plus directement et principalement sur les connaissances. Comme le formule Manuel Castells (1998), "pour la première fois dans l'histoire, l'esprit humain est une force de production directe, et pas simplement un élément décisif du système de production".

Au terme de cette analyse, le logiciel apparaît ainsi comme un objet technologiquement complexe : il vise à apporter des solutions à des problèmes de plus en plus complexes – que dans certains cas il contribue à complexifier davantage – en étant partie prenante d'un système, le système informatique, qui se caractérise également par une grande complexité. Il existe une dynamique puissante d'augmentation simultanée de la puissance des matériels et de la complexité des logiciels, basée sur la force des relations de complémentarité techniques et économiques entre ces deux activités.

Malgré sa jeunesse, le logiciel est marqué par un foisonnement d'innovations, qui concernent l'extension accélérée de ses domaines d'applications, la création de nouveaux produits, le renouvellement des méthodes et des techniques de production, et l'utilisation d'outils d'automatisation (génie logiciel). La forte dynamique innovatrice de l'économie du logiciel est confirmée par le fait que les innovations dans le domaine des logiciels concernent *tous* les types d'innovations mis en évidence par les économistes qui souhaitent appréhender

l'ensemble des innovations existant dans l'économie. Par contre, les rythmes de diffusion très différenciés de ces innovations expliquent l'hétérogénéité technologique de l'économie du logiciel et le fait qu'elle semble caractérisée par une instabilité technologique permanente.

Sur un plan plus économique, le développement de logiciels sur mesure constitue une activité de services, alors que les progiciels sont des biens intangibles. Les progiciels possèdent "naturellement" des caractéristiques de biens collectifs des biens intangibles. De ce fait, pour rendre un progiciel appropriable, il est nécessaire de recourir à des dispositifs techniques (à l'efficacité limitée) et à des dispositifs juridiques. La variété de ces dispositifs, les problèmes que pose leur application aux progiciels, expliquent la diversité des statuts juridiques des progiciels et l'apparition de formes originales comme les licences publiques à la base des logiciels libres. Enfin, les logiciels se caractérisent par des mécanismes singuliers de détermination de leur prix. Les théories de la valeur ne semblent pas pouvoir s'appliquer aisément au cas des logiciels : il est particulièrement difficile d'appréhender la valeur d'usage d'un logiciel, ce qu'illustre particulièrement les débats sur l'appréciation des effets de l'informatique ("paradoxe de Solow"). Quant à la théorie de la valeur travail, elle ne peut s'appliquer à des biens qui, soit ne sont pas reproductibles (logiciels sur mesure), soit sont trop facilement reproductibles (progiciels).