UNE TECHNOLOGIE QUI CONTINUE DE PROGRESSER

# L'accumulation du logiciel : de la programmation à « l'informatique dans les nuages » *[cloud computing]*

Le mot logiciel a été créé en 1973 pour désigner l'ensemble des programmes et procédés utilisés pour traiter des données au moyen de matériels informatiques. Cette activité, qui semblait simple et logique, s'est révélée beaucoup plus complexe à maîtriser que prévu…

par **Jean-Paul FIGER***

Cet article retracera les principales étapes de l'évolution du logiciel, puis il évoquera des perspectives pour l'avenir.

Les progrès de la miniaturisation – voir la célèbre loi de Moore (1) – fixent le rythme d'évolution de la technologie du matériel. Cela étant, c'est la vitesse de déploiement du logiciel, qui est l'élément déterminant pour la pénétration des ordinateurs dans toutes les activités, tant intellectuelles qu'industrielles. Le logiciel transforme l'ordinateur, d'outil capable, en théorie, de résoudre un problème, en un outil qui le résout en pratique. Le matériel est au logiciel ce que les instruments sont à la musique, que Léonard De Vinci définissait comme le « modelage de l'invisible ». Cette définition s'applique peut-être encore mieux au logiciel…

Le graphique ci-après (voir figure 1) présente la répartition de la dépense informatique en Europe, en 2006. La part du logiciel est prépondérante, alors que celle du matériel est tombée à moins de 20 %.

Si les progrès ont été foudroyants dans le domaine du matériel, ils ont été tout aussi impressionnants dans le domaine du logiciel. Quinze ans après la définition de l'architecture des machines par Von Neuman (en 1945), presque tout avait déjà été inventé en matière de logiciel, et les programmeurs étaient déjà dans l'attente de machines plus puissantes pour continuer à progresser. Il suffit pour s'en convaincre, de se souvenir des dates d'apparition des langages informatiques successifs : le FORTRAN en 1957, le LISP en 1959, le COBOL en 1960 et le BASIC en 1964. Il en va de même en ce qui concerne les systèmes d'exploitation.

* Président, ARMOSC
http://www.figer.com


(1) Gordon E. Moore, cofondateur d'Intel, avait prédit en 1965, que le nombre de transistors des microprocesseurs doublerait tous les deux ans.

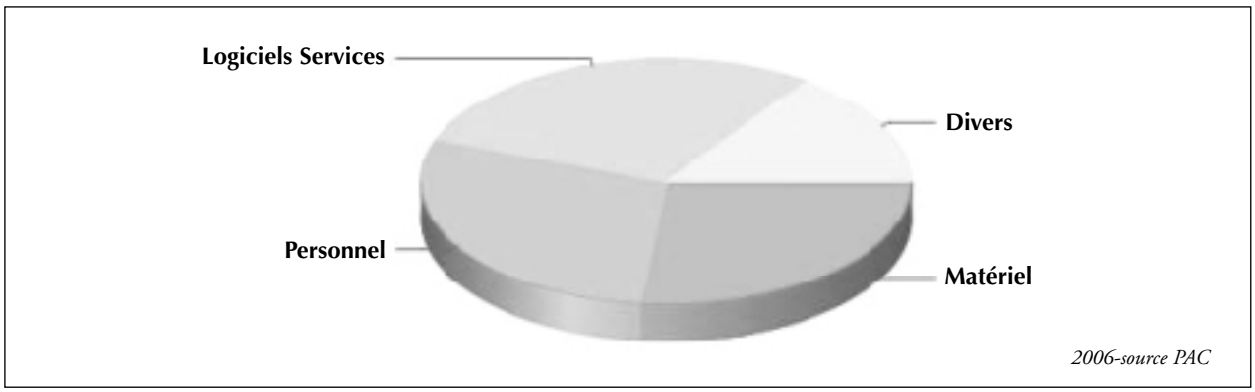


**Figure 1.** Répartition de la dépense informatique en Europe en 2006.

Les fonctions dont pouvait disposer un programme développé en 1966 sous l'O.S. (*Open Space*) des ordinateurs IBM 360 étaient pratiquement les mêmes que celles qui sont aujourd'hui disponibles. La première version d'UNIX a quant à elle été développée dès 1969 et les bases de données relationnelles sont nées en 1970.
Les premiers programmes d'ordinateurs avaient été conçus par des mathématiciens et des scientifiques qui pensaient qu'il s'agissait là d'un travail simple et logique. Mais le logiciel s'est avéré plus difficile à développer qu'ils ne l'avaient supposé : les ordinateurs étaient têtus ; ils s'obstinaient à faire ce qui était écrit, plutôt que ce qui était voulu. Le résultat fut qu'une nouvelle corporation d'artisans prit le relais, pour accomplir cette tâche. Bien souvent ni mathématiciens, ni scientifiques, ils étaient profondément engagés dans une aventure avec les ordinateurs, une aventure qui préfigurait une nouvelle science.

## L'ORIGINE DES LANGAGES DE PROGRAMMATION

L'idée des langages de programmation est aussi vieille que celle du calculateur numérique. Il suffit d'essayer, ne serait-ce qu'une seule fois, de programmer un ordinateur directement en code binaire pour comprendre immédiatement pourquoi… Les premiers programmeurs en sont donc arrivés rapidement à inventer des notations symboliques, appelées « langages » et traduites en code binaire par un programme appelé « compilateur ». Parmi les premiers langages, celui dont l'influence a été la plus grande est sans nul doute le FORTRAN [*FORmula TRANslation*], développé entre 1954 et 1957 par John Backus et ses collègues de la société IBM. Il n'était pas sûr qu'un compilateur puisse produire du code d'une manière efficace, à une époque où la puissance des machines était encore très limitée. Mais cet objectif fut atteint et le FORTRAN est toujours utilisé aujourd'hui.
Cependant, la version originale de FORTRAN comportait des contraintes inutiles, des structures de données limitées et, surtout, de graves déficiences dans le contrôle de la logique du programme. En un sens, on peut dire que toutes les recherches entreprises afin de définir de nouveaux langages de programmation ont été motivées par des tentatives de résoudre les défauts inhérents au langage FORTRAN.
Que les langages informatiques aient été définis par un comité (comme le COBOL), une organisation commerciale (comme le PL/I), un individu (comme PASCAL) ou encore sous l'égide du Ministère de la Défense américain, le *DoD* (comme ADA), toutes les tentatives engagées pour définir le Langage universel ont échoué, laissant la voie libre à des milliers de langages, dont une petite dizaine, seulement, sont largement répandus.
Contrairement à ce qui s'est passé pour les matériels, les progrès accomplis dans le domaine du logiciel ne proviennent pas d'une seule technologie, ni même d'une technologie dominante. Dans le domaine des langages, les progrès résultent de meilleures structures de contrôle des programmes, de meilleurs environnements de programmation et d'outils de programmation plus puissants. L'évolution semble lente, mais les progrès sont effectifs. Après quelques années, ces progrès ne sont plus perçus comme de simples améliorations, mais comme de réelles nouveautés. Ce qu'il y a de surprenant, c'est le fait que les vieilles techniques ne disparaissent pas. Certains continuent de programmer dans des langages datant de plus de cinquante ans, comme le FORTRAN, voire même en usant d'une écriture ancienne connue sous la dénomination d'assembleur ; pour d'autres, ces outils apparaissent comme de véritables fossiles vivants ! Charles Quint disait : « Je parle en espagnol à Dieu, en italien aux femmes, en français aux hommes et en allemand à mon cheval ! » Apparemment, il en va pour les machines, comme pour les hommes : il n'existe pas, non plus, de langage informatique idéal…

### L'évolution des langages de programmation

Le langage machine des premiers ordinateurs était fortement contraint par les capacités du matériel : il fallait loger des instructions machines dans le plus petit nombre possible de bits, si l'on voulait obtenir de bonnes

performances. Par exemple, il y avait des instructions de branchement qui différaient selon la longueur du saut. Cela n'était déjà pas très facile à gérer, pour un programmeur ; pour un compilateur, c'était quasi impossible. Le bricolage était donc la règle. La première révolution a été lancée par Edsger W. Dijkstra (1930-2002), dans une lettre publiée en mars 1968 dans les communications de l'ACM (2), intitulée *Go To Statement Considered Harmful.* Il y prônait l'abandon du *GO TO* dans tous les langages de programmation de haut niveau, au profit de structures de contrôle évoluées, alors dénommées « programmations structurées ». Il avait remarqué que la qualité des programmes était inversement proportionnelle au nombre de *GO TO* ! Il a quand même fallu attendre plus de vingt ans, pour que cette évidence soit reconnue par tous et mise en pratique dans les langages, tout d'abord, puis dans les programmes.

Après la percée des structures de contrôle, l'enjeu sur les langages s'est reporté sur les types de données. Les premiers langages de haut niveau avaient très peu de types de données : nombres entiers, tableaux, chaînes de caractères, etc. Le raisonnement dominant était le suivant : en multipliant les types de données afin d'en contraindre le contenu (c'est-à-dire, par exemple, en déclarant qu'une chaîne donnée de caractères correspond à une date, à une couleur ou à une adresse Internet), le compilateur sera à même de détecter un maximum d'erreurs pendant la phase de compilation, avant le passage à l'exécution. Les programmes deviendront donc plus simples à mettre au point et nombre d'erreurs disparaîtront. L'exemple type de cette démarche a été celle du Ministère de la Défense américain (le *DoD*), qui voulait imposer un langage de programmation unique pour toutes les applications militaires de l'informatique. Ce fut le langage ADA, développé par un Français, Jean Ichbiah (1940-2007), qui gagna la compétition. A l'usage, on s'est aperçu que les langages fortement typés exerçaient une contrainte trop forte sur les programmeurs, sans apporter les bénéfices que l'on pouvait en attendre au niveau de la mise au point. ADA n'a fait que s'ajouter à la longue liste des langages utilisés par le *DoD*, sans en remplacer un seul, après quoi, il a disparu. Ce fut une des dernières tentatives de création d'un langage unique « bon à tout faire » (3).

## L'approche « objet » ouvre la voie à la réutilisation

Dans les langages procéduraux (FORTRAN, COBOL…), les instructions du langage sont impératives et s'adressent à une entité non désignée, tout simplement parce qu'une seule possibilité s'offre : l'ordinateur considéré dans sa globalité. Dans les langages « orientés objet », l'ordinateur est virtuellement divisé en objets, qui peuvent être « adressés » individuellement. Ces objets peuvent communiquer entre eux, en s'envoyant des messages. Cette approche a été introduite, pour la première fois, par Ole-Johan Dahl et Kristen Nygaard dans Simula 67, un langage dérivé d'Algol 60. L'idée n'a pas suscité beaucoup d'intérêt jusqu'au développement de *Smalltalk*, en 1970, par Alan Kay et ses collègues du *Xerox Palo Alto Research Center*. Cette technique était devenue indispensable pour programmer des systèmes comportant des fenêtres et utilisant la « souris ». Cette approche s'est répandue progressivement durant les années Quatre-vingts (Objective C, C++), au fur et à mesure que les performances croissantes des ordinateurs masquaient l'inévitable inefficacité du code.

Avec les langages procéduraux classiques, le logiciel est produit en enchaînant des procédures qui contiennent des types de données et des algorithmes. La programmation « orientée objet » généralise la notion de type, combine algorithmes et données en objets et supprime la décomposition fonctionnelle des langages procéduraux. Cette décomposition est remplacée par trois concepts : l'héritage, le polymorphisme et l'échange de messages. L'héritage permet de définir de nouveaux objets à partir d'objets existants. Le polymorphisme permet de définir des actions indépendantes des caractéristiques d'un objet particulier : la fonction « IMPRIMER », par exemple, peut être valide aussi bien pour du texte que pour des images, et ce, aussi bien vers une imprimante noir et blanc que vers une imprimante couleurs. A la place de gros blocs de programmes contenant des clauses « *if-then-else »*, un programme « orienté objet » contient des fonctions polymorphes, qui résolvent les *« if-then-else »* au cours de l'exécution. Cette souplesse a toutefois un prix : les temps d'exécution des programmes sont notablement plus longs (de deux à dix fois).

Par ailleurs, l'héritage et le polymorphisme sont tout aussi complexes et causes d'erreurs que la décomposition fonctionnelle traditionnelle. Ainsi, l'approche « orientée objet » n'apporte guère de gains en termes de productivité des programmeurs.

En revanche, cette approche a d'autres mérites : en particulier, elle introduit la notion de réutilisation. Avec la réutilisation, l'heure des projets informatiques partant de zéro est révolue. Les concepteurs capitalisent sur ce qui fonctionne, afin de l'améliorer et de l'enrichir. Cette tendance a démarré (au sein d'universités américaines, comme le MIT) sous l'impulsion des communautés de *hackers,* ceux qui exploitent au maximum les propriétés des logiciels existants. Ces communautés se sont surtout développées avec l'arrivée d'Internet, qui a permis aux développeurs du monde entier de conjuguer leurs efforts. L'exemple emblématique est le système d'exploitation Linux, dont la démarche de développement pragmatique, menée sur Internet par un petit nombre

(2) http://www.cs.utexas.edu/users/EWD/ewd02xx/EWD215.PDF

(3) Java, pensé comme langage « bon à tout faire », est, je l'espère, la dernière tentative du genre. Lourd à écrire, gourmand en mémoire et d'une exécution lente, il fait le bonheur des vendeurs de matériels. Les utilisateurs l'ont chassé de leur poste de travail : il ne survit plus que sur les serveurs des entreprises.

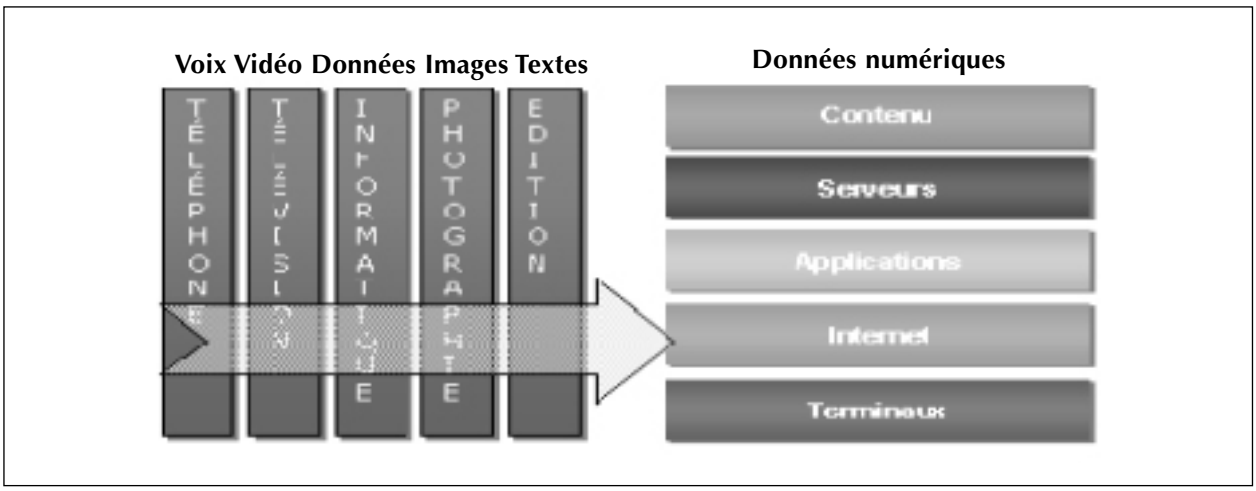


**Figure 2.**

de programmeurs motivés, a progressivement balayé les UNIX propriétaires. C'est de cette mouvance qu'est né le mouvement Logiciel Libre [*Open Source*].

## Internet change la donne

Le *DoD* (15 % de la dépense informatique mondiale, à la fin des années Soixante-dix) a raté son objectif d'imposer un langage unique dans le domaine des langages de programmation. En revanche, il a connu un très grand succès dans le domaine des réseaux d'ordinateurs, en créant l'Internet (4). De 1 000 ordinateurs connectés en 1984, on est passé à 10 000 en 1987, à 100 000 en 1989, à 1 000 000 en 1992 et à plusieurs centaines de millions en 2008, pour une population d'utilisateurs d'environ 1,3 milliard. Plus de la moitié des ménages français disposent d'un accès Internet haut débit, c'est-à-dire d'un accès à l'ensemble des connaissances mondiales (pour le meilleur ou pour le pire) et, ce, pour un forfait d'environ 30 euros par mois. Cette infrastructure mondiale unique, qui se met en place autour des réseaux de calculateurs reliés par l'Internet, aura sans doute une plus grande influence sur notre société que celle de tous les réseaux d'infrastructure précédents (canaux, chemin de fer, autoroutes, téléphone…) (voir figure 2).

Autrefois, chaque média avait à construire sa propre infrastructure spécialisée. Pour fournir un service de téléphonie, il fallait d'abord faire des investissements considérables dans un réseau dédié et dans des équipements terminaux. Pour développer la télévision, il avait été nécessaire de tout standardiser, des caméras jusqu'aux récepteurs de télévision. Ces deux réseaux avaient été, en leur temps, des réussites techniques exemplaires. Cependant, l'interdépendance de tous les éléments rendait les évolutions très difficiles et très lentes. Le passage du téléphone au numérique (avec Numéris) a été un échec commercial, car il nécessitait de changer les équipements terminaux, pour un avantage minime. L'évolution de la télévision vers la couleur, ou le son stéréo, a été rendue lente et complexe par la nécessité de rester compatible avec le parc des téléviseurs installés. De surcroît, les opérateurs qui contrôlaient l'infrastructure n'avaient aucune incitation à suivre la baisse des coûts des équipements informatiques. Par exemple, la réduction considérable des coûts des calculateurs (le rapport est de l'ordre de 1 à plusieurs millions !) ne s'est pas vraiment traduite par la baisse des prix des communications téléphoniques. En fait, l'intégration verticale et l'interdépendance entre toutes les couches permettaient aux fournisseurs de facturer un coût global incluant le coût de la communication et celui des services fournis.

L'infrastructure Internet change donc la donne, en introduisant la concurrence à tous les niveaux. C'est ce qui permet, par exemple, à Google de fournir gratuitement une messagerie, ou à Skype d'offrir un service de téléphonie mondiale gratuit, à partir d'une infrastructure existante.

Surtout, et c'est le plus important, cette nouvelle infrastructure favorise l'innovation. Les nouvelles applications ou les nouveaux services peuvent bénéficier, sans investissement supplémentaire, d'une connectivité mondiale, à destination de plus d'un milliard de clients potentiels et, ce, pour un coût dérisoire.

Ce marché d'un milliard d'utilisateurs, accessible quasi gratuitement au travers d'une simple liaison Internet, devient l'enjeu de la Ruée vers l'or du XXIe siècle : la bataille des parts de marché dans la nouvelle économie, autour des domaines clés que sont : les portails d'accès, les services financiers ou la vente au détail. Les valorisations boursières atteintes sont à la mesure des espoirs suscités.

## La révolution du Logiciel Libre (Open Source)

Moins visibles, mais beaucoup plus importants pour l'évolution de l'informatique, ce milliard d'utilisateurs pèsent sur la manière de développer et de distribuer les

(4) http://fr.wikipedia.org/wiki/Histoire_d %27Internet

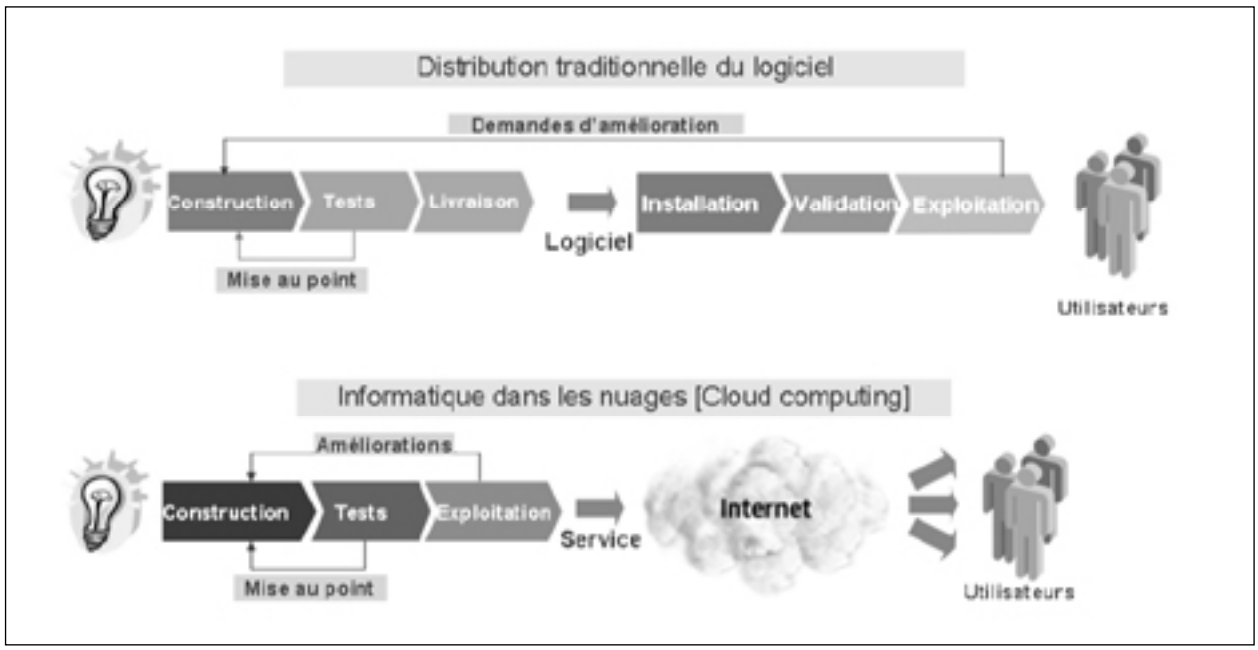


**Figure 3.** Distribution traditionnelle du logiciel.

logiciels. Toutes les innovations technologiques et tous les nouveaux développements sont passés au crible de ce milliard de clients, expérimentateurs ou développeurs. Un processus de sélection, qu'on pourrait qualifier de darwinien, remplace les décisions de comités trop souvent biaisées par des considérations politiques, voire l'incompétence de leurs membres. Cette formidable coopération mondiale informelle a déjà produit les meilleurs logiciels (ceux à la base du succès d'Internet). Elle a imposé ses choix, comme le MP3 pour la diffusion de la musique, et elle a fédéré, sous la bannière LINUX, tous les UNIX propriétaires. L'époque où un fournisseur pouvait protéger son « empire » au moyen de normes propriétaires est désormais révolue !
La révolution du Multimédia bouscule les acteurs en place. Sa maîtrise nécessite de changer profondément la manière de communiquer, de gérer l'information, de faire du commerce, de travailler, de concevoir et de réaliser des produits. Le risque est grand de continuer à faire du vieux avec du neuf. En évitant les biais des réseaux précédents (qui étaient optimisés autour d'un contenu), Internet a créé une place de marché pour des concepts nouveaux qui tirent parti d'une connectivité abondante et bon marché. Cette infrastructure représente la frontière clé entre un média de communication et les applications construites sur ce média ; c'est une frontière très spéciale, puisqu'elle permet à ces deux marchés de fonctionner chacun selon ses propres règles. Avec les nouvelles technologies numériques, l'avantage s'est déplacé vers une conception souple, qui permet de tirer rapidement parti d'opportunités non prévues au départ.

## « L'informatique dans les nuages » (cloud computing)

Internet et les logiciels libres ont favorisé la création de logiciels interopérables, performants et très peu coûteux. Cependant, le modèle de distribution est resté inchangé, avec la nécessité d'installer une copie du logiciel sur chaque machine. Cette distribution du logiciel, simple, a priori, est devenue complexe, en raison du très grand nombre de logiciels installés sur chaque machine. Avec plus de 200 logiciels utilisés sur mon PC, la mise à jour et l'application des correctifs de sécurité deviennent une activité journalière. J'ai même dû installer un logiciel de plus, chargé de tout surveiller !
Une connectivité mondiale, assise sur des standards précis, va permettre de bouleverser ces pratiques.
Un nouveau modèle « à la Google » est, en effet, en train de s'imposer. Programmes et données sont hébergés sur des infrastructures mondiales dimensionnées largement. L'utilisateur accède à son environnement de travail et à ses applications au moyen d'un navigateur standard, sans qu'il soit nécessaire d'installer préalablement des logiciels. Cette idée était vieille comme l'informatique (le *time-sharing* des années Soixante). Ce qui est nouveau, et qui change tout, c'est la performance, mais également la modicité du coût. Dans la plupart des cas, la performance est supérieure à celle obtenue avec des logiciels installés en local. Les coûts sont en général très bas, puisqu'on ne paye que ce que l'on utilise. La mise en route est immédiate et il n'y a plus de mises à jour à gérer. Chaque matin, on découvre de nouvelles fonctions. Un autre avantage, déterminant, est la possibilité d'accéder à ses données et applications favorites, à partir de multiples terminaux, fixes ou mobiles, sans avoir à les copier, ni à les synchroniser.

La perte d'une machine (vol ou détérioration accidentelle) n'entraîne plus aucune perte de données. Il est aussi possible d'avoir accès à des applications telles que Google Earth, dont le volume des données – rien moins que la cartographie mondiale – rend impossible un stockage en local.
Ce vaste marché est en train de s'organiser, allant de pures offres d'infrastructure – comme Amazon S3, pour le stockage ou EC2, pour la puissance machine – jusqu'à des offres de services applicatifs, tels que la gestion de photos (FlickR), de vidéos (YouTube) ou d'outils de bureautique (Google Apps).

## Et le matériel devient lui-même logiciel

La préservation des investissements colossaux réalisés dans le domaine des logiciels a été une préoccupation constante des utilisateurs, et ce dès les débuts de l'informatique. Pour durer, un constructeur devait garantir que les programmes déjà écrits continueraient à fonctionner sur les nouveaux matériels : c'est la compatibilité ascendante, laquelle a assuré la domination de la gamme IBM 360 ou du microprocesseur Intel 8086 et suivants. Cette compatibilité ascendante complique considérablement les matériels et constitue un frein à l'innovation. Les performances des techniques dites de « virtualisation » et la simulation du fonctionnement d'un matériel sur un autre ont beaucoup progressé. Longtemps réservées au développement des systèmes, elles sont aujourd'hui couramment utilisées dans des environnements de production. Il n'est pas rare que l'on fasse ainsi fonctionner plusieurs dizaines de machines, avec des configurations matérielles et logicielles différentes, sur un seul matériel physique. Les avantages sont nombreux : économies de temps – une nouvelle machine peut être créée par simple copie de fichiers – réduction du gaspillage tant en termes de matériel que d'énergie (car les taux d'utilisation des machines sont en général faibles), simplification de l'exploitation. Il est aussi possible de continuer à utiliser des matériels qui n'existent plus. J'ai commencé à programmer en 1962, en utilisant le langage FORTRAN, sur un ordinateur IBM 1620. Si j'avais conservé les programmes que j'ai écrits à l'époque, je pourrais continuer aujourd'hui à les utiliser sur mon PC, grâce à une machine virtuelle, disponible sur Internet (5).

## L'avenir du logiciel

Les limites physiques de la miniaturisation des composants des matériels informatiques vont être atteintes dans les dix prochaines années. L'amélioration – exponentielle – des performances des matériels que nous avons connue depuis cinquante ans, et qui a tiré le développement de l'informatique, va donc se ralentir fortement. Seuls les réseaux, grâce à la fibre optique, conservent encore aujourd'hui une marge de progression importante. Ce sont donc le logiciel et les réseaux, qui seront les sources essentielles des prochaines innovations. Nul doute que les millions de programmeurs motivés, connectés entre eux par l'Internet, produiront des logiciels extraordinaires.

(5) http://www.jowsey.com/java/sim1620/ConsoleApplet800.html