

# L'exclusion des programmeurs autodidactes

PAR FRÉDÉRIC MAURE

L'informatique, d'abord comme service spécialisé des entreprises, puis comme activité des sociétés indépendantes, a été pendant vingt ans une voie permettant aux autodidactes d'acquérir un statut professionnel. L'industrialisation progressive de la production de logiciel, avec ses exigences de méthode et de profondeur technique, ferme progressivement cette voie à ceux qui n'ont pas de formation ni de diplôme.



A la fin des années 60, les services informatiques étaient en train de se structurer dans les grandes entreprises. Leur position était floue sur les organigrammes, tantôt annexe de la Comptabilité, tantôt service de la Direction du Personnel. Les descriptions de poste de pupitreur, d'exploitant, de programmeur étaient relatives aux habitudes locales, et de toutes les façons, il n'y avait pas de diplôme d'informatique.

A l'époque, quand vous étiez un débrouillard sans diplôme, il vous était facile de commencer par devenir monteur de bandes en exploitation, puis pupitreur, et d'exploitant passer programmeur au gré des occasions. Elles ne manquaient pas : la croissance de l'informatique était brutale.

Pour vos compétences techniques minimales, les constructeurs pourvoyaient aux notions de bases de leur Cobol. Il fallait tout au plus quelques semaines à IBM ou Control Data - précurseur de l'enseignement privé - pour transformer le titulaire d'un BEP astucieux en gardien du temple de la programmation. Ceux qui avaient fait de vraies études allaient éventuellement vers l'analyse, mais un peu d'expérience en programmation ouvrait aussi bien la porte.

C'était aussi l'époque où les centres de calcul et les services de programmation étaient des navires aux équipages soudés par les rythmes du travail en équipes, où l'exploitation était aussi noble que le développement, et où l'atelier de traitement imposait ses priorités aux programmeurs. Pour accéder à ces métiers, il ne s'agissait pas d'être un gros têtard, mais bien de savoir vivre leurs rythmes. On demandait juste d'avoir *l'esprit logique*, ce qui signifiait plutôt savoir suivre une procédure pas à pas que manifester des capacités d'abstraction.

## Savoir transposer ses connaissances

En 1990, les passerelles ouvertes aux autodidactes vers des statuts nouveaux disparaissent. Programmeur et autodidacte sont aujourd'hui des notions contradictoires pour les directions informatiques.

Elles ont un certain nombre d'arguments techniques. Les langages utilisés sont de plus en plus variés, alors qu'il y a quinze ans une bonne pratique du Cobol et du JCL local suffisait à tenir plusieurs années. Aujourd'hui, à chaque type d'application son langage, son progiciel ou sa base de données. Savoir transposer ses connaissances à de nouvelles versions est une exigence implicite.

Il serait bien sûr possible aux programmeurs de se former. Mais l'essentiel des formations offertes sont celles des constructeurs. Liées aux achats de l'entreprise, elles sont essentiellement de l'entraînement à la syntaxe des langages et aux commandes des systèmes. Il est long et coûteux de former aux notions de fond dont pourtant seule la maîtrise permet l'utilisation des fonctions complètes des outils d'aides à la programmation, à commencer par les compilateurs. C'est que la complexité et le gré d'abstraction des notions sur lesquelles reposent ces outils du logiciel ont beaucoup progressé. Quel autodidacte fera le saut conceptuel d'un modèle de données Merise, de spécifications écrites en Ada, d'un langage de commande engendré par un petit automate abstrait ?

## Ingénieurs maison

Les directions informatiques et les sociétés de services spécialisées s'appuient aussi sur de nouvelles conceptions des rapports de travail pour exclure les autodidactes. Parmi les obligations de moyens pour réaliser un logiciel, les compétences sont difficiles à décrire dans un contrat autrement qu'en référence aux diplômes des développeurs. Combien d'ingénieurs maison parmi les programmeurs ? La sous-traitance n'est pas la seule à être dans ce rapport avec les services utilisateurs, c'est-à-dire avec les payeurs des développements d'applications. Dans l'entreprise même, les méthodes d'assurance - qualité modélisent peu ou prou les relations entre services internes sur les rapports entre des clients et des fournisseurs. L'informatique est dans le rôle du fournisseur, soumis aux exigences du client, exigence répercutée sur la qualification des programmeurs.

La corporation a de son côté peu à peu produit des critères d'accès à ce qui est devenu un métier. Les pionniers ont vu leur exigence professionnelle reconnue, mais un directeur informatique de 1990, même ancien pupitreux des années 70, n'embauchera plus en dessous de DUT. Les rémunérations se sont de leur côté structurées autour des diplômes : BTS, DUT, MIAGE, Maîtrises, DESS et DEA, et si les non-diplômés peuvent encore entrer dans le métier, ils sont alors ouvertement sous-payés.



Disegno di MARCO VAGLIERI

## Des passages qui se ferment

Pourtant, la Faculté a toujours du mal à situer l'informatique parmi les sciences. Technologie, métier de savoir-faire, voire ingénierie de compromis et d'imperfection, la programmation semble contradictoire avec ses fondements théoriques logiques et rationnels. Les cours sur la théorie ne produisent pas d'opérationnalité mais les TP et les stages sont à eux seuls insuffisants pour faire comprendre un style de programmation. Les diplômes d'informatique sont plus que dans d'autres disciplines des repères oscillants, et des services de recrutement tiennent à jour leurs listes des bonnes et des mauvaises écoles.

Une fois les autodidactes des premiers temps calés dans les statuts de managers par effet de génération, l'informatique actuelle offre-t-elle encore des tangentes, des passages, des transgressions sociales parmi les métiers et les filières professionnelles ?

Pour les logiciels techniques, proche du système, soumis à de fortes contraintes de méthodes de programmation, c'est déjà fini. Hors une solide formation, point de salut. Dans l'informatique de gestion, la diffusion progressive de méthodes d'analyse de plus en plus formalisées se répercute sur des programmeurs. Aux analyses reste la capacité à connaître le terrain des applications, connaissance informelle, intelligence des situations et des relations.

Les programmeurs d'aujourd'hui qui n'ont pas l'habitude des représentations formelles butent sur les multiples codes qu'utilise maintenant le métier pour décrire les problèmes et les procédures. Plus que des connaissances supplémentaires, que les autodidactes sont surentraînés à accumuler, ces langages, avec les formalismes précis et les méthodes de pensée qu'ils structurent, sont de plus en plus hors de portée de ceux qui ne les ont pas acquis au départ.