COBOL : l'étonnante jeunesse, avec SQL et Java

Le langage COBOL toujours sans successeur

Pierre Fischof, Consultant

Il peut sembler paradoxal pour une grande quantité d'esprits scientifiques et des affaires que le langage COBOL né en 1959, dont on annonça la mort proche presque à sa jeunesse, soit resté encore en pleine vigueur pour écrire les grandes applications de gestion.
Presque sans successeur mais non sans évolutions avec l'état de l'art technologique le plus récent ni sans combinaisons avec d'autres langages.

Existe-t-il une alternative à COBOL, SQL et Java ?

Fig.1: Grace Murray Hopper au pupitre du UNIVAC, c. 1960 (wikipedia anglophone)

Y a-t-il un secret à la longévité si surprenante du langage COBOL ?
Oui, certainement.
On pourrait sans doute résumer à quatre les raisons principales à la source de la fontaine de jouvence dont jouit le langage COBOL depuis tant d'années :

- L'**adaptation** : un projet collectif au départ parfaitement adapté par ses promoteurs aux besoins finals ;
- **Ses promoteurs** : lesquels sont puissants, motivés, unis et dynamiques ;
- **L'évolution** : Un projet suivi et adapté de façon collective, ceci au travers six standards successifs ;
- **L'absence d'un successeur** : qui soit à la fois crédible et apte à le remplacer.

De plus, des féministes ajouteront que c'est une femme, Grace Hopper, qui est l'une des principales initiatrices de ce langage, au travers du langage FLOW-MATIC en 1954, à côté du langage COMTRAN de Bob Bemer en 1957, comme les rapportent les encyclopédies Wikipédia francophone et anglophone.

On peut noter aussi que le suivi du langage lui-même fut le fait d'organismes mixtes.
L'Armée américaine, si innovante et pragmatique, reconnaissons-lui cette qualité, en matière informatique, avait besoin d'un langage de gestion universel, manipulant aisément les fichiers, fiable et simple à documenter et à maintenir. Avec différents partenaires, elle prit donc l'initiative de construire le langage COBOL, lequel fit l'objet rapidement d'un consensus international industriel et économique, voire scientifique, pour pouvoir développer les grandes applications de l'informatique de gestion. C'était le langage dont toutes les grandes administrations semblaient avoir le plus grand besoin.

Il est à noter que l'on devra plus tard aussi à l'Armée américaine la création d'un magnifique langage industriel nommé ADA, mais dont le génie scientifique, si proche du francophone Pascal, n'eut d'égal que son peu de succès à l'exportation, hors une utilisation interne parfaitement efficiente pour construire des systèmes fiables ; ceci malgré des mérites scientifiques incontestés.

Les atouts initiaux du COBOL

C'est peu après l'utilisation massive et incontestée des langages d'assemblage, proches des instructions d'une machine dédiée et presque directement traduisibles en code exécutable binaire, que naquirent les langages développés universels scientifiques et de gestion que furent FORTRAN et COBOL, indépendants des constructeurs de machine et de leurs systèmes d'exploitation.

Le coût de l'heure de machine commençant à baisser relativement par rapport à celui des heures d'ingénieurs développeurs et aux coûts de maintenance des applications, on souhaita confier donc à la machine l'effort de traduction des instructions d'un langage plus proche de l'homme en code assembleur puis en code directement exécutable par la machine.

Le principal atout de COBOL était, outre la puissance grandissante de ses instructions, plus proches de l'homme par rapport à l'assembleur, l'aisance de son auto-documentation et la clarté de sa structure, certes contraignante au départ, mais obligeant à déclarer toutes les données et leur structure hiérarchique, dans des sections dédiées, avant-même leur utilisation par la procédure.

On peut dire que ce sont ces qualités mêmes d'exigence théorique de maintenabilité qui l'ont souvent fait détester par les amoureux de la virtuosité informatique et du bidouillage astucieux et libre. Ces derniers ont donc préféré des langages mieux adaptés à leurs désirs.

Les renouvellements incorporés aux langages

Grâce à ses promoteurs internationaux et à leur union, COBOL a su se renouveler au travers de six standards successifs validés par l'ANSI (American National Standards Institute), intégrant, même avec retard, les éléments de l'état de l'art en matière technologique et des langages de programmation.

- **COBOL-60**, le premier d'entre eux, définit un langage intelligible par l'homme, simple à comprendre et à documenter, pouvant appeler des sous-programmes tant écrits en COBOL qu'en assembleurs ou en d'autres langages, et spécialisé dans la manipulation simple de nombreux fichiers et structures de données complexes ;
- **COBOL-68** incorpore la gestion automatisée des tables et tris automatisés, la segmentation possible des programmes pour leur gain de place en mémoire, les différents modes d'accès directs aux fichiers, les facilités de programmation d'éditions automatisées ;
- **COBOL-74** poursuit les améliorations et corrige des défauts du langage qui compliquaient sa maintenance. Comme il a intégré de façon modulaire (par les instructions EXEC et END-EXEC), la gestion des différents moniteurs interactifs de transactions et des différentes bases de données du marché, il intégrera plus tard de la même façon le nouveau langage SQL d'accès aux bases de données relationnelles qui l'enrichira ainsi ;
- **COBOL-85** adopte la programmation structurée comme mode de programmation possible, avec les séquences répétitives et alternatives et leurs bannières de fin de séquences intégrées au langage (END-IF, END-PERFORM, END-READ), le transformant en langage algorithmique, assouplissant sa structure et sa syntaxe et l'affranchissant des anciennes colonnes figées et des points à chaque instruction ;
- **En 1989**, le langage intègre les fonctions intrinsèques, mathématiques, de chaînes de caractères ou autres, sans besoin de recours à des sous-programmes externes écrits en Fortran ou en C, par exemple,
- **COBOL-2002 (COBOL Objet)** adopte et incorpore la programmation objet, avec tous ses mécanismes d'héritage, l'ouverture au standard XML de définition des données, la gestion des pointeurs, des booléens, des bits ainsi que le support de toutes les langues nationales et leur alphabet au moyen d'Unicode…

À côté de ces standards, certains acteurs n'hésitent pas à créer leurs propres extensions au langage COBOL, en lui adjoignant des instructions de programmation des interfaces Web, comme ce fut le cas, par exemple, de l'éditeur de logiciel Micro Focus.

Beaucoup de langages prometteurs

L'apparition de nombreux nouveaux langages scientifiquement « géniaux » n'a pas manqué de voir le jour depuis la création de COBOL.
On peut citer parmi ceux-ci Algol, Pascal et ADA, lesquels, pour la déception de beaucoup, ne reçurent pas le plein succès que méritaient leurs très grandes qualités.
À côté de nombreux langages propriétaires plus ou moins liés à un constructeur, d'autres langages universels, plus proche du système, apparurent, tels C, C+, C++, qui permirent, avec les systèmes d'exploitation universels Unix et Linux, de concevoir des systèmes et applications portables indépendants de la machine.
De nombreux autres langages plus spécialisés, tels HTML, PHP, Python et Javascript remplirent parfaitement et efficacement la tâche à laquelle ils étaient dédiés.
Cependant, parmi les autres langages que COBOL ayant réussi à s'imposer universellement au milieu des années 2010, il semble qu'il y a surtout le SQL, pour la gestion des Bases de Données Relationnelles, et Java pour l'écriture des interfaces homme-machines et l'adaptation à la technologie Web, pour régner en maîtres incontestés sur toutes les plates-formes.

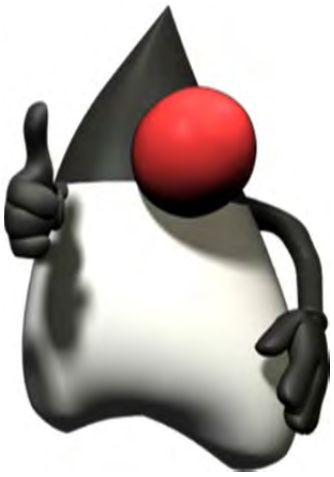

Fig. 2 Duke, la mascotte Java (Wikipédia)

Les promesses dangereuses des marchands du temple

C'est à la fin des années quatre-vingt et dans les années quatre-vingt-dix que des industriels et financiers intéressés, suivis de nombreux professionnels et journalistes peut-être un peu naïfs, promirent à la fois la fin des gros ordinateurs et de leurs fournisseurs, et celle sans appel du langage COBOL. Ces oracles annonçaient, comme une nouvelle collection de mode ou une révolution culturelle, le triomphe immédiat de la décentralisation informatique, au nom du mot d'ordre de « down-sizing », et l'avènement et la généralisation de la technologie et des langages dits « Client-Serveurs», que peu de personnes et peu d'organisations étaient humainement capables de maîtriser, malgré les progrès technologiques survenus.

À court terme, l'opération connut certes pour ses promoteurs un succès financier intéressant.
À long terme, en revanche, de nombreuses entreprises utilisatrices durent déchanter et revenir en arrière vers des choix informatiques plus robustes, éprouvés et maîtrisables, à moins de disparaître. Les sommes fabuleuses englouties dans de tels projets furent alors passées le plus souvent par pertes et profits.

Quelques années après, dans les années 2000, revint alors la mode de la centralisation et de la consolidation informatique et comptable des groupes et de leur fédération autour de progiciels culturellement uniformisants tels SAP, parallèlement à une poursuite de la déconcentration et départementalisation de l'informatique.

La situation surprenante des applicatifs planétaires

Aujourd'hui, si l'on s'en réfère aux enquêtes reconnues du Gartner Group, 75 % des données du monde des affaires sur la planète étaient toujours traitées en 2005 par des programmes écrits en COBOL ! (80 % en 1997 avec 200 milliards de lignes de code).

Cette situation, surprenante en apparence, montrerait la différence entre la partie visible et la partie immergée de l'iceberg des systèmes d'information automatisés. Notons que l'épineux passage à l'an 2000 des programmes imparfaitement codés nous a fourni une excellente occasion d'inventaires assez exhaustifs du patrimoine applicatif.

Pour ce qui concerne les développements futurs, en revanche, le Gartner Group estimait que 15 % des nouveaux programmes développés le seraient encore en COBOL, ce qui est déjà une grosse quantité… (en 1997, 5 milliards de nouvelles lignes de code écrites annuellement en COBOL étaient estimées).

Dernièrement, une importante administration française choisissait, comme beaucoup d'autres entreprises, de développer ses nouveaux projets les plus sensibles en COBOL, SQL et JAVA, ceci à l'aide du recours temporaire à des compétences externes, et décidait de former méthodiquement ses chefs de projet à ces langages pour la maintenance.

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO-WORLD.

PROCEDURE DIVISION.
DISPLAY 'Hello, world'.
STOP RUN.
```

Exemples de déclaration de données en Cobol (http://en.wikibooks.org/wiki/COBOL)

```
77 I pic s9 (4) usage is binary.

01 TRANSACTION-RECORD.
 05 RECORD-NUMBER PICTURE S9 (7)
 COMP-3 VALUE ZERO.
 05 RECORD-DESCRIPTION PICTURE X
 (30) VALUE SPACES.
 05 EDITED-AMOUNT PIC
 $$$$,$$$,$$$.99-.
 05 FILLER PIC X (60) VALUE
 SPACES.
```

Exemples d'ordres de calcul et d'affections en Cobol (http://en.wikibooks.org/wiki/COBOL)

```
MOVE A TO B.
COMPUTE GROSS-PAY = HOURS-WORKED
* HOURLY-RATE
MULTIPLY HOURLY-RATE BY HOURS-
WORKED GIVING GROSS-PAY
SET MY-INDEX TO 1

MOVE A TO B. *> THIS IS A
COMMENT ON THE SAME LINE AS A
STATEMENT
```

Exemples de structures alternatives en Cobol (http://en.wikibooks.org/wiki/COBOL)

```
If my-number is numeric
      continue
else
      display 'data field "my-
      number" is not numeric'
end-if
```

Exemples de structures répétitives en Cobol (http://en.wikibooks.org/wiki/COBOL)

```
Perform 6 times
      add +1 to loop-count
end-perform

Perform process-1-billing-record
until all-records-processed

Perform clear-1-table-entry
varying tbl-index from +1 by +1
until tbl-index is greater than
max-table-entries
```

Pouvoir développer durablement

Reste le problème que, malgré l'avènement de nombreux langages candidats en eux-mêmes très méritants, peu de ceux-ci ont réussi à faire consensus en dehors d'un champ d'application spécialisé.
En conséquence, aucun langage crédible n'est encore apparu pour reconquérir la place du vétéran COBOL afin de pouvoir le supplanter et lui succéder…

Le plus paradoxal n'est-il pas que pour principales raisons invoquées qui rendraient prétendument nécessaire un abandon du langage COBOL, l'on évoque une certaine pénurie de ressources humaines actuelles et d'étudiants formés au langage, en particulier en France ?

Si les écoles, universités et industries ont manqué d'esprit visionnaire et de pragmatisme en sous-estimant cette formation stratégique, invoquer cette raison serait-il légitime pour freiner l'utilisation du langage ?
Ne serait-il pas un argument à courte vue ?

Reste aussi la question : cela serait-il souhaitable et quelles seraient les raisons authentiques qui rendraient pertinentes le remplacement d'un langage plutôt que le choix de l'évolution de celui-ci ?
Cette question ne mérite-t-elle pas d'être posée ?▲

pierre.fischof@yahoo.fr

Notes

On trouvera dans la version anglophone de Wikipedia ainsi que dans ses annexes Wkibook et Wikiversity maintes riches informations sur les langages qui ne figurent pas encore dans la version francophone.