COBOL: l'histoire d'un langage qui refuse de disparaître

Stéphane FOSSE
fosse.fr

15 décembre 2025

Plus de soixante ans après sa naissance, COBOL fait tourner les systèmes financiers de la planète. Ce langage créé dans l'urgence au crépuscule des années 1950 devait résoudre un problème simple : comment écrire une fois un programme pour qu'il fonctionne sur différentes machines ? Personne n'imaginait alors qu'il survivrait jusqu'au XXI^{e} siècle.

À la fin des années 1950, le traitement des données commerciales réclamait un outil adapté. Les entreprises manipulaient des volumes croissants d'informations, mais chaque fabricant d'ordinateurs imposait son propre langage. Un programme écrit pour un IBM ne fonctionnait pas sur un UNIVAC. Cette fragmentation coûtait cher et ralentissait le développement des applications administratives qui commençaient à proliférer.

Trois projets distincts tentaient déjà de résoudre ce casse-tête. FLOW-MATIC, développé pour l'UNIVAC, résultait du travail mené par Grace Murray Hopper sur la série A des ordinateurs algébriques et scientifiques. Ce langage opérait sur les machines UNIVAC I et II dès décembre 1956. Le Commercial Translator d'IBM, confronté à des conflits juridiques autour de son nom COMTRAN, représentait une autre approche. AIMACO, conçu par l'Air Materiel Command à Dayton dans l'Ohio, complétait ce trio de prétendants.

Mai 1959 : la rencontre qui changea tout

Le 28 mai 1959, une cinquantaine de personnes se réunissent au Pentagone à Washington. Charles A. Phillips, à la tête du Data Systems Research Staff, a rassemblé militaires, fonctionnaires, consultants et ingénieurs des plus grands fabricants d'ordinateurs. La rencontre révèle le développement séparé de trois langages similaires. L'exemple d'ALGOL justifie la tentative de bâtir un langage commercial commun.

De cette rencontre naît le CODASYL, un comité chargé de réfléchir aux langages informatiques. Un groupe plus restreint, baptisé « comité court terme », reçoit la mission d'analyser les points forts et les faiblesses des compilateurs existants. Joe Cunningham de l'Air Force en prend la vice-présidence. Gene Albertson d'U.S. Steel, Greg Dillon de DuPont et Mel Grosz d'ESSO rejoignent le comité directeur avec les présidents des trois groupes de travail créés à cette occasion.

Wegstein dirige le groupe Short Range, répondant aux fondateurs d'ALGOL du National Bureau of Standards. Gene Smith du Bureau of Ships supervise l'Intermediate Range. Bob Curry de Southern Railway conduit le Long Range. Grace Hopper et R.W. Bemer sont désignés comme conseillers techniques. Personne ne revendique l'acronyme « COBOL ».

Le groupe Short Range se voit confier la tâche de produire une proposition en trois mois. Certains dans l'ancienne garde rient en apprenant cette échéance. Les représentants viennent de Burroughs, IBM, RCA, Minneapolis Honeywell, Remington Rand UNIVAC et Sylvania, auxquels s'ajoutent l'Air Materiel Command et le Bureau of Ships. Entre le 23 juin et la fin août, ils tiennent douze réunions.

Le 4 septembre, le rapport est présenté au Comité Exécutif. Wegstein déclare charitablement qu'« il contient des points approximatifs et nécessite quelques ajouts ». Le document forme un ensemble hétéroclite qui pousse IBM à abandonner ses plans pour Commercial Translator. Le groupe reçoit pour mission de le mettre en forme d'ici le 1er décembre.

Contre toute attente, personne ne s'attendait à ce que ce comité accouche d'un nouveau langage. En novembre 1959, six personnes s'enferment pendant deux semaines. Ils en sortent avec les spécifications d'un langage qu'ils nomment COBOL : Common Business Oriented Language. Ce document, finalisé en avril 1960, va bien au-delà d'une simple synthèse des langages existants.

Le choix technique du COBOL repose sur sa structure en quatre divisions. IDENTIFICATION contient les informations sur le programme, ENVIRONMENT décrit l'environnement d'exécution, DATA structure les données et PROCEDURE contient les instructions. La syntaxe se rapproche de phrases en anglais, rendant le code lisible y compris pour des non-spécialistes. Les noms de variables comptent jusqu'à 30 caractères, une nouveauté qui rend les programmes compréhensibles sans recourir à des abréviations cryptiques.

Les résistances d'IBM et la bataille des standards

Malgré les inadéquations du rapport, la liste des fabricants annonçant leur engagement ou leur intention d'implémenter COBOL s'allonge. La question reste simple : qu'allait-on faire ? Beaucoup de doutes subsistent sur un langage défini en grande partie par des exemples plutôt que par une syntaxe et une sémantique rigoureuses. IBM, en particulier, s'oppose à l'adoption du langage. John Backus avait présenté sa notation métalinguistique l'été précédent. Charlie Katz de General Electric, autre représentant d'ALGOL qui avait trouvé un écho dans l'annonce publique du langage GECOM, affirme que COBOL pourra être accepté par les GECOM s'il se développe au point où une interprétation ambiguë devient impossible.

La position officielle d'IBM sur COBOL représente un facteur déterminant pour l'industrie. Commercial Translator avait été annoncé pour les 7070, 709/90 et 705 III. Barry Gordon assume la responsabilité des implémentations des compilateurs. Roy Goldfinger et R.W. Bemer travaillent tous deux au sein d'IBM et au sein du Comité Short Range pour réduire les différences entre Commercial Translator et COBOL.

L'annonce du 27 janvier 1960 indique qu'IBM inclura le COBOL de base dans Commercial Translator. L'enquête du 15 février par l'organisation SHARE montre Commercial Translator comme la version COBOL d'IBM, avec 80% d'indépendance machine. Honeywell décline de citer FACT car il concerne une seule machine, mais Wegstein revendique 100% pour COBOL.

Lors de la réunion SHARE de février, Al Harmon, responsable de la Programmation Appliquée, affirme qu'« il semble que les calendriers de réalisation d'une version de COBOL satisfaisante pour tous les ordinateurs existants et proposés retarderaient indûment la production par IBM de processeurs pour Commercial Translator ». L'entreprise révise sa présentation du meilleur effort pour résoudre ces problèmes. La position officielle d'IBM, annoncée dans le magazine Datamation, précise que « le Commercial Translator est retravaillé aussi proche que possible de l'esprit COBOL pour garantir que le résultat final sera un langage unique utilisable pour le traitement de données ».

Joe Cunningham rapporte en mars 1963 que, lorsque le nettoyage total a été réalisé, la syntaxe est aussi définie que celle d'ALGOL, mais la sémantique reste sujette à des ambiguïtés. Le vrai problème tient au fait qu'il existe deux versions de Commercial Translator au sein d'IBM. Tom Glans, Roy Goldfinger et R.W. Bemer spécifient une version à fusionner dans COBOL, ou du moins à réconcilier avec son identité. L'autre version est écrite par Barry Gordon et diverge. Roy Goldfinger réalise une comparaison extensive en juillet 1960 entre COBOL et la version de Commercial Translator implémentée par Barry Gordon, montrant que l'objectif initial a été manqué de loin.

IBM annonce finalement sa production de compilateurs COBOL en octobre 1960, mais il faut attendre septembre 1962 pour que le succès de COBOL devienne suffisamment apparent et que Bob Ruthrauff dise à SHARE XIX qu'« Nous n'avons pas l'intention de faire de COBOL notre langage de développement et ne planifions aucun développement supplémentaire du langage Commercial Translator lui-même ».

Lors de la réunion CODASYL du 7 avril 1960, NCR et General Electric annoncent leurs intentions de construire des compilateurs COBOL. En mai, Jack Jones de l'Air Materiel Command annonce le démarrage d'un COBOL UNIVAC 1105. Dès septembre, onze fabricants sont représentés sur CODASYL et six ont indiqué qu'ils fourniront des compilateurs.

Le désaccord interne chez IBM ne cesse de croître. Le problème est transmis à T.V. Learson, alors président du conseil d'IBM, qui le résout dans le style de quelqu'un qui dispose d'un argent dépensable différent. IBM fera les deux et travaillera vers la réconciliation. Al Harmon informe la réunion du 17 mai 1960 que GUIDE, un autre groupe d'utilisateurs des ordinateurs IBM, a voté qu'IBM fournirait des compilateurs pour le 705 II sans IOCS, le 705 III, 7080, 7070 et 709/90, mais refuse de donner des dates de livraison. Cette approche ne satisfait pas GUIDE, qui résout qu'IBM devrait s'en tenir à sa déclaration initiale et qu'il ne devrait y avoir qu'un seul compilateur par machine.

IBM ne fait rien de tel. Non pas parce qu'ils ne le veulent pas, mais parce qu'ils ne le peuvent pas. Le compilateur 7090 pour Commercial Translator, réalisé par un groupe de la côte Ouest sous la direction du Dr. Richard Talmadge, est si performant comparé aux compilateurs COBOL existants à l'époque qu'en février 1963, SHARE abandonne tout espoir pour Commercial Translator et prépare la transition des programmes existants vers COBOL via des routines de traduction et quelques manipulations manuelles.

Décembre 1960 : la démonstration qui fit basculer l'industrie

Le New York Times du 26 août 1960 annonce la victoire de RCA dans la « Course des Traducteurs Informatiques », bien que le langage ne soit pas pleinement COBOL selon les moyens de l'époque. Le compilateur est publié en décembre. En novembre, RCA et UNIVAC organisent une démonstration de compatibilité commune pour le public.

Les 6 et 7 décembre 1960 survient le moment de vérité. Un même programme COBOL s'exécute correctement sur deux machines radicalement différentes, un RCA 501 et un UNIVAC II. UNIVAC II le 6 décembre et RCA 501

le 7 décembre. Pour la première fois, la portabilité du code informatique devient réalité. Le magazine Datamation annonce que RCA « fait de la publicité capitalisée sur le Succès Sacré » et que l'entreprise a « tiré profit des premières billes de publicité ». IBM reçoit une publicité similaire avec le compilateur COBOL 61 sur le 1410.

Les techniques de compilation des premiers processeurs COBOL sont primitives, ce qui se traduit par des taux de compilation très faibles. Les évaluations de la Navy rapportées en mai 1962 montrent cinq compilateurs allant de 3 à 11 instructions par minute. Les mesures de mi-1964 révèlent une plage de 11 à 1000 instructions par minute. Le matériel ne devient pas beaucoup plus rapide en deux ans, il faut donc découvrir comment construire de meilleurs compilateurs. À ce moment, on remarque pour la première fois comment le taux de compilation varie drastiquement avec la taille de la mémoire disponible et comment la variance de coût est importante. Cette étude couvre une fourchette de 0,23 à 18,91 dollars par instruction.

Les premières années de COBOL sont marquées par plusieurs évolutions. COBOL-61, COBOL-61 Extended, puis COBOL-65 se succèdent. L'intention initiale vise une mise à jour annuelle, mais celle-ci entre en collision avec l'avènement de la standardisation formelle début 1963. La normalisation ANSI arrive en 1968, suivie par d'autres normes en 1974 et 1985. La compatibilité ascendante, garantissant que les anciens programmes continueront à fonctionner, est une caractéristique remarquable qui sera maintenue.

Le comité COBOL de CODASYL se concentre sur le développement tout en laissant la standardisation intermédiaire et formelle aux organismes de normalisation. L'European Computer Manufacturers Association crée le Comité Technique 6 pour traiter COBOL, correspondant au Comité X3.4.4 de l'American National Standards, qui assume la responsabilité de produire une norme COBOL. Howard Bromberg en est le président. Ces deux comités travaillent conjointement pour produire douze Bulletins d'Information COBOL.

Sous le Brooks Bill, Loi Publique 89-306 des États-Unis, le Federal Information Processing Standards définit la responsabilité tripartite du National Bureau of Standards, du Bureau of Budget et du General Services Administration. Le premier homme à occuper le poste au NBS est Norman Ream, précédemment de Lockheed. Le financement restreint au NBS empêche l'embauche et la mesure des efforts programmés initialement. Ces difficultés sont surmontées lorsque Ream est nommé Assistant Spécial au Secrétaire de la Navy, car il est convaincu que les normes ne seront pas trop contraignantes sans un moyen de vérifier leur conformité.

De nombreux compilateurs prétendent être COBOL, mais sans une loi Truth In Packaging ou un dispositif de détection, l'utilisateur se trouve parfois trompé. La solution ingénieuse de Ream consiste à demander au Commandant de Service Grace Hopper de l'USNR à la retraite de diriger l'accélération de la standardisation COBOL pour la Navy. Grace doit retourner au service actif pour six mois à partir du 1er août 1967. Une seule chose ne va pas dans le mémo qui annonce ce changement : elle ne prend pas sa retraite en 1968, ni en 1969, ni en 1970, lorsque ses ordres sont réédités pour une durée « indéfinie ». La Navy dispose maintenant de certifications COBOL complètes et exhaustives disponibles pour tous.

Par son intégration sous la standardisation de contrôle, COBOL tire profit de sa relation avec d'autres normes de traitement de l'information. CODASYL a présenté un premier rapport sur la réunion du 13 janvier 1961 de l'American Standards Association, lors de laquelle le Comité X3, consacré aux Ordinateurs et au Traitement de l'Information, a été autorisé. Les jeux de caractères occupent une place très importante dans la discussion. FLOW-MATIC dispose de 63 caractères disponibles en raison de l'ordinateur UNIVAC, alors qu'IBM est limité à 48 à cause de la carte perforée de l'époque. Une spécification COBOL est adoptée par l'International Standards Organization. L'ECMA contribue largement avec une description formelle de la syntaxe de COBOL, qui se révèle précieuse pour réduire les ambiguïtés et valider les constructions.

Le gouvernement américain exige que tout ordinateur vendu ou loué aux administrations dispose d'un compilateur COBOL. Cette décision force les constructeurs à développer leurs propres compilateurs, propulsant COBOL au rang de standard incontournable dans l'informatique de gestion. La longévité extraordinaire de ce langage né comme une solution temporaire s'explique par plusieurs éléments. Sa relative simplicité le rend accessible aux programmeurs non académiques qui forment le gros des équipes informatiques des entreprises. Sa lisibilité facilite la maintenance des applications. Son orientation vers le traitement des données administratives correspond parfaitement aux besoins du secteur tertiaire.

Face au bug de l'an 2000, le monde découvre avec stupeur l'omniprésence de COBOL dans les rouages de l'économie. Les estimations vertigineuses montrent que sur les 300 milliards de lignes de code en production mondiale en 1997, environ 240 milliards sont écrites en COBOL. Plus de 95% des données financières et d'assurance transitent par ces programmes.

Contre toute logique apparente, COBOL traverse le cap de l'an 2000 sans perdre son importance. En 1999, plus de 50% des nouvelles applications critiques sont encore développées dans ce langage, face à l'émergence de Java. Les projections pour 2004-2005 estiment que 15% des nouvelles applications (5 milliards de lignes) seront en COBOL, tandis que 80% des applications déployées constitueront des extensions à des programmes COBOL existants.

Cette résistance inattendue trouve sa source dans les caractéristiques techniques du langage : pas de pointeurs, des types de données élémentaires, une description détaillée des fichiers et des sorties d'impression. Ces limitations, qui semblent handicapantes pour des applications scientifiques, sont en réalité des atouts pour les applications de gestion en réduisant les risques d'erreurs catastrophiques.

Nous voici en 2025 avec un univers COBOL qui défie tous les pronostics de disparition. Plus de 220 milliards de lignes de code COBOL alimentent encore les systèmes financiers mondiaux, et contrairement aux prédictions des années 1990, ce volume continue même de croître de plusieurs milliards de lignes chaque année. Une structure organisationnelle existe pour le développement et plusieurs organismes de normalisation encadrent les changements et les ajouts au fur et à mesure qu'ils se développent.

Plusieurs facteurs créent néanmoins un équilibre précaire. La pénurie de développeurs s'aggrave, avec 68% des spécialistes COBOL qui devraient partir à la retraite d'ici 2025. Cette raréfaction des compétences transforme paradoxalement le langage en opportunité lucrative : un expert COBOL facture désormais entre 150 000 et 200 000 euros par an. Les entreprises recherchent activement des profils hybrides, capables de faire le pont entre les systèmes legacy COBOL et les technologies modernes comme Java ou Python.

La question de la modernisation reste centrale. Les institutions qui ont évalué la charge de réécriture de leurs applications COBOL arrivent à une estimation moyenne d'une dizaine d'années de travail, sans compter les risques critiques. Les systèmes COBOL coûtent 20 à 25% plus cher à maintenir que les environnements modernes, mais les tentatives de migration se révèlent catastrophiques, comme l'ont démontré les déboires de la banque TSB en 2018 ou de la Commonwealth Bank of Australia, dont la migration a nécessité cinq ans et coûté près d'un milliard de dollars australiens.

Des experts de la qualité logicielle affirment que COBOL demeure plus sûr et plus performant que des langages plus récents comme Java, notamment parce que ces systèmes ne sont généralement pas exposés à Internet et qu'ils ont été scrutés par des générations d'informaticiens dans une industrie où la sécurité reste primordiale. Les banques ont peaufiné ces programmes pendant des décennies pour qu'ils s'exécutent avec une efficacité remarquable en termes de débit et de rythme transactionnel.

Bien que le bug de l'an 2049 évoqué dans certaines discussions ne concerne pas directement COBOL lui-même (qui ne possède pas de type date natif), les systèmes informatiques qui l'utilisent devront gérer le bug de l'an 2038, où les timestamps Unix sur 32 bits atteindront leur limite le 19 janvier 2038. Tout comme pour l'an 2000, les entreprises devront investir massivement dans des corrections, sans pour autant abandonner leurs systèmes COBOL.

L'industrie tente de s'adapter. Une version récente, COBOL V6, vient d'être publiée avec des exécutables optimisés, et le langage recommence à être enseigné dans les écoles de développeurs, même s'il reste rarement le choix numéro un en sortie d'école. Des initiatives comme GnuCOBOL émergent pour moderniser le langage. Les entreprises adoptent désormais des méthodes agiles pour les projets COBOL, avec des sprints d'environ trois semaines, abandonnant les cycles en V traditionnels.

Le concept de niveaux dans les standards COBOL risque de devenir source de tension. Si l'échange fait l'objet d'une pression trop forte, la tendance ira vers l'écriture de programmes au niveau le plus bas, afin de garantir la plus grande utilisabilité sur un nombre maximal d'ordinateurs, chassant ainsi les programmes de haut niveau selon une logique qui rappelle la loi de Gresham.

COBOL et les autres langages possèdent tous une faille sérieuse dans l'usage public des données : ils transportent la description des données dans le programme. Pour l'échange et l'usage multiple de données publiques, les données devraient être auto-descriptives, avec leur description transportée sur le support de stockage lui-même.

Ces difficultés techniques ne devraient pas décourager l'utilisateur d'une utilisation intensive du langage COBOL. Les problèmes de transférabilité des programmes et les pertes résultant de langages non transférables dépassent largement le milliard de dollars chaque année. COBOL continue d'être utilisé dans de nombreux secteurs critiques : banques, assurances, administrations, transport aérien, et même l'industrie spatiale et automobile. Tant qu'il y aura des systèmes financiers traitant des milliards de transactions quotidiennes, COBOL conservera sa place dans les infrastructures critiques mondiales.

Dans notre monde obsédé par l'innovation perpétuelle, la simplicité et la stabilité l'emportent sur la sophistication. Un langage jugé dépassé par les informaticiens depuis les années 1970 continue de faire tourner les systèmes financiers mondiaux au XXI^{e} siècle, démontrant que l'élégance théorique n'est pas toujours gage de pertinence pratique.

Références

[1] American National Standards Institute. American National Standard X3.23, COBOL. Federal Information Processing Standards Publication FIPS PUB 21-1. National Institute of Standards et Technology, 1974.

[2] R. W. Bemer. A View of the History of COBOL. In : *Honeywell Computer Journal* (). Reproduit de M. Berk's *The Programmer's COBOL*, Inter-ACT/McGraw-Hill.

[3] Julien Chrétien. COBOL : Le langage informatique oublié qui fait encore tourner le monde. Mars 2025.

[4] CodeAura. Pourquoi la modernisation du COBOL est essentielle pour l'avenir du secteur bancaire. Mars 2025.

[5] IDG News Service. Les banques restent fidèles à Cobol, plus performant que Java. Le Monde Informatique, juin 2013.