2512COBOLnow.pdf
2025.12 ブログ：『COBOL の今昔』の詳細
（→ http://www.1968start.com/M/blog/index4.html#2512b ）

ＣＯＢＯＬの今昔

中所武司

■このエッセイのきっかけ

忘れかけていた COBOL の下記の記事を読んで、懐古趣味的に
４ＧＬ（第４世代言語）、Ｙ２Ｋ（西暦 2000 年問題）を思い出した。

・朝日（2025.12.8）：
どっこい生きてる「ＣＯＢＯＬ」　大型コンピューターのプログラミング言語
https://digital.asahi.com/articles/DA3S16359459.html

■記事の要約とコメント（→★）

・企業や官公庁の基幹システムの大型コンピュータで必要な
プログラミング言語 COBOL の技術者のニーズが見直されている。

・分散・小型化で大型コンピュータはなくなるといわれたが、多くの企業で使われ続け、
技術者の多忙な状況を取材すると、ＩＴ業界が抱える課題がみえてきた。

官公庁・企業の中核に残る「古く信頼性ある技術」
６０・７０代エンジニア現役　若手不足、モチベ課題

・９月まで金融機関のシステム開発現場で働いていたＩＴ技術者（58 歳）の話：
メインフレーム（大型コンピュータ）が COBOL によって動いていた以前の職場には
74 歳と 71 歳の技術者もいた。知識が豊富なので働き続けてこられたのだろう。

・COBOL はコンピュータの初期から使われている。英単語を使うため、直観的に理解しやすい。
バージョンアップによる仕様変更が少なく、技術者が高齢化しても扱える。

→★拙著（共著）「プログラミングツール」（昭晃堂 1989 年）には、下記の記載あり：
https://www.1968start.com/M/book/progToolCH4.pdf
４章　プログラミング
4.2　プログラミング言語
4.2.2 手続き型高級言語
(2) COBOL
【抜粋ｐ.104】
『1959 年に・・・開発されたもので、1968 年に標準規格化された。
比較的簡単な構文仕様で、事務処理プログラムを英語（自然語）風に記述できる。
計算機の利用目的では事務処理が圧倒的に多いということもあり、
COBOL は世の中で最も広く使われている言語である』

・一方、プログラムは長くなりやすく、技術者の技術力の違いも出やすい。
改修を重ねて、改修の記録が残っていないと、プログラムの解読困難で、
改修できないレガシーシステムになるリスクもある。

→★前世紀の終わりに大きな社会問題となった「西暦 2000 年問題」（Ｙ２Ｋ）を思い出す：
COBOL の技術者が少なく、修正が間に合わないことが危惧された。

拙著「ソフトウェア工学」（第３版 朝倉書店 2014 年）では、下記記載：
https://www.1968start.com/M/lecture/SE3index.html
2.　ソフトウェア工学の概要
2.2　ソフトウェア工学の課題
2.2.5　ソフトウェア生産技術の課題
【抜粋ｐ.12-13】
『・・・20 世紀末の西暦 2000 年問題であった．
これは，プログラムの中で西暦を下二桁で表現して処理しているシステムが多く存在し，
2000 年になった時に 1900 年として扱うために誤作動を引き起こすという問題であった．
この問題は第一義的には**質の問題**であった．その故障によって生じる
社会的影響の大きな情報システムが多数存在していることを顕在化させる結果となった．
次に，修正すべきシステムが多数存在するにもかかわらず，その保守に短期間に対応できる
技術者の数には限りがあるという意味では**量の問題**でもあった．
また，10 年以上前に開発されたメガステップオーダの大規模ソフトウェアの中から，
まともな保守書がないまま変更箇所を探し出し，適切な修正を加える作業は
規模の問題でもあった．
さらに，ネットワークを介した電子商取引の時代には，一つの誤った送信データが
他の企業や他国の情報システムを混乱させる可能性をはらんでいたという意味では
インタフェースの問題であった.』

・今の小型コンピュータでは記号や関数を使ってコンパクトにプログラムをつくれる。
また、既製のソフトを組み合わせることでシステムを効率的に作ることもできる。

→★80 年代の第４世代言語（４ＧＬ）も「コンパクトにプログラムをつくれる」だった。
拙著「ソフトウェア危機とプログラミングパラダイム」（啓学出版 **1992 年**）に下記記載：
https://www.1968start.com/M/keigaku/sp11.pdf
第 11 章　エンドユーザコンピューティング
11.4　業務向け簡易言語
【抜粋ｐ.211】
『業務向け簡易言語として第４世代言語（４ＧＬ）がある。第４世代言語という名前は、
第１世代言語（機械語）、第２世代言語（アセンブラ）、第３世代言語（コンパイラ言語）に
続く言語という意味で付けられた。
James Martin は、４ＧＬを 13 ヶ条の特徴で定義したが、その主なものは、
非職業的プログラマが使用可能、アプリケーションプログラムの記述ステップ数と
作成工数が COBOL より一桁少ない、非手続き的記述形式の採用、などである・・・』

・大卒で、ウェブサイトやウェブアプリの作成経験もある技術者（26 歳）は 6 月に、
エンジニア 200 人余りの COBOL 専門の会社に転職した。前職では技術者の競争が激しく、
画面表示のわかりやすさなどのセンスを問われる面も大きかった。

・これに対して COBOL はルールがはっきりしていて、正確さが求められる。
設計書通りにプログラムを作って、求められた通りの結果を出さないといけないので、
最初は『お堅い仕事』だと思った。

・COBOL 専門会社での先輩の技術者（41 歳）話では、COBOL 自体は覚えることが少なく、
顧客がシステムで実現したい業務をいかにプログラムで実現するかが問われるので、
言語そのものより、オーダーメイドで作り上げていくことが重要とのこと。

→★how-to-make よりも what-to-make が重要ということですね。
ユーザの要求仕様を確認しながら実装することで、手戻りを防ぐということ。

・手書きでプログラムを書いていたころから COBOL を使ってきた技術者（67 歳）は、
定年退職後の再就職先の COBOL 専門会社で最高齢。
COBOL はなくなると言われ続けたので、サービスを提供する会社も技術者が足りない状況だ。
大型機に代わるシステムは簡単にはでてこない。70 歳までは働くつもりと話している。

・経済産業省の研究会は 2018 年、複雑化した「レガシーシステム」の障害で、
年間１２兆円もの損失がでる可能性を「2025 年の崖」と表現して警告した。
メインフレームを知る技術者が退職・高齢化することも要因としてあげた。

・メインフレームは、1960 年代に企業のホストコンピューターとして広がった。
日本 IBM が作った市場に、日立製作所や富士通などの大手電機メーカーが参入した。

・90 年代には比較的安価なサーバーやパソコンが使われるようになり、ダウンサイジングで
いずれメインフレームはなくなるという見方が広がった。

・富士通は、製造・販売を 1930 年度で、サポートは 1935 年度までに終了と 2022 年に公表。
今年度末で 540 台の稼働を見込む。システムを入れ替えるために３～４年かかることもあり、
新たなメインフレームを提供しながら進めている。
日立製作所はすでに製造をやめ、ユーザには IBM 製品を自社ブランドとして提供する。

→★富士通は小規模企業向けのオフコンに関しても撤退を決めている。
参考ブログ：2025.11「オフィスコンピュータの今昔？」
https://www.1968start.com/M/blog/index4.html#2511

・COBOL 専門会社の社長によると、同社設立の 20 年前に 20 万人前後いた COBOL の技術者は、
１万人程度までに減少。COBOL 技術者の平均年齢は 50 代後半になった。
若い技術者が極端に不足しているが、需要は確実にあるとのこと。

・IBM が 11 月に東京で開いた「若手技術者カジュアル・ミートアップ」では、
約 50 人の参加者は、取引先企業 30 社の 9 年目までの技術者で、200 人を超えた。

・「若い社員は将来性のない部署に配属されたと不安になる。実際は顧客の基幹業務の担当。
中核のシステムを把握することになる」と、事業部長は仕事の重要性を指摘する。

・メインフレームが使われる領域は減ったが、企業のシステムの中核部分では残り続けている。
三菱 UFJ 銀行は「メインフレームの処理性能と信頼性は不可欠」と位置づける。
IBM とも協力して、地銀 20 行が参加する「共同プラットフォーム」を運営する。

・官公庁や流通業、製造業など幅広くを提供してきた NEC も
「メインフレームで培ってきたＩＴ資産を生かすためにも、
サーバーやクラウドとの共存を前提に、不可欠な技術だと考えている」と位置づける。

・経産省は５月のリポートで、企業経営者らに「レガシーシステムからの脱却」を呼びかけ、
メインフレームを「古い技術」などと位置づけた。
ＡＩ産業戦略室は「古くて信頼性がある技術なので、保守できれば動く。
ただ、シェアの大きな国産ベンダー（業者）に続けられないところもでている」と説明。

→★COBOL プログラムをブラックボックス化してサブシステムとして使い続けるとしても、
今の時代に合わせて、複数のサービス（サブシステム）の連携を容易に可能とするような
インタフェースは必要と思われる。

以上