中央大学経済研究所年報　第46号（2015）pp.479-495

COBOL資産のオフショアマイグレーションによるソフトウェア産業の活路について

鳥 居 鉱 太 郎

1980年代から続く数十万人といわれるソフトウェア開発要員不足の問題は，1990年代に入って海外に開発の拠点を設けるオフショア開発の需要へと展開している。アジアのオフショア開発は2000年代以降も人材供給という点で開拓が進んでいるが，成熟期を迎えた国から人件費の高騰がはじまり，インフラ環境，政治・経済情勢などを考慮した上で，人材の取り合いとなる難しい局面を迎えている。本論は，50年以上の変遷を経て培われたメインフレーム（大型汎用機）におけるソフトウェア資産が，そのオープン化においてオフショア開発の諸問題を解決する新たな原動力となりえることを示すものである。

1．はじめに

国内におけるソフトウェア開発要員の不足は，オフショア開発によっても人件費の高騰や離職の問題から解決には至っていない。高度ICT人材の育成として，オフショア委託先でシステム開発の下流工程のみでなく上流工程に対応する人材育成をするも，より高収入を目指した離職の問題をかかえる傾向にある。すなわち，オフショア開発の主な動機である人材確保や開発コスト削減の両面において，大きな問題をかかえているのが現状である。オフショア開発がまだ進んでいない国で新規の委託先を開拓しても，経済社会の成熟過程において同様の現象が起きることは明らかで，こうした時間差の問題で凌ぐだけでは，日本におけるソフトウェア開発の人材不足に関する解決には至らないと考えられる。

ここで，数十年に亘って蓄積されてきたソフトウェアの資産という観点から，こうした問題を解決する新たな可能性を見出すことができる。ソフトウェア開発ではプログラミング言語が用いられるが，近年になって新規開発されたシステム以外は，1960年代から主にCOBOL[1)]（コボル：COmmon Business Oriented Language）が使用されている。基幹業務

1）米国CODASYL（Conference on Data Systems Languages）によって，1959年までに開発された事務処理用プログラミング言語。1960年には最初の仕様であるCOBOL-60が発行された（Bermer, R. W. (1971), “a view of The History of COBOL,” *Honeywell Computer Journal*, Vol. 5, No. 3, pp. 132-133.）。

の多くはメインフレームやオフコン上のシステムとして開発され，運用されている。こうしたシステムは一般には直接目にすることがなく，膨張するスマートフォンのアプリ市場やWebシステムなど，直接に利用者として接することもない。したがって一般にはレガシーなシステムとして過去の遺物のように誤解されるケースがあるが，高い信頼性が求められ安定運用されるメインフレームのシステムは，極めて重要な社会基盤を形成しているのである。一般的に，オフショア開発先の国々においてレガシーシステムの歴史は日本や欧米と異なり浅く，またそれに対応する技術の蓄積や人材に乏しい。COBOLによるシステムの資産は50年以上にわたる蓄積から，そのプログラム量は極めて膨大である[2)]。そして，それらは高度に安定運用されているため，レガシーシステムを捨てて同様のシステムを新規開発する動機は通常起こりえない。ここにオフショア開発の展開において新たな可能性があることを提案するものである。

国内の膨大なCOBOL資産は，システムすべての移行案件という意味では大規模開発が一巡した後に激減するはずであるが，オープンシステムへの資産継承という観点からは，依然として資産としてのCOBOLシステムは引き継がれることになる。ここでレガシーシステムの技術の蓄積を持つ国内ベンダーと，基幹システムおよびその運用の技術を求めるオフショアベンダーとの間に，共通の意識を見出すことができる。この関係はレガシーシステムの歴史がないオフショア先の諸国間では離職の問題を抑制し，下流工程による請負関係から，上流工程も含むすべてのフェーズに関与する高度なICT人材育成ならびにパートナー関係を可能にする要素となる。

2．COBOLの特徴と資産的な重要性

2-1 開発言語の変遷およびCOBOLを取り巻く状況

1940年代以降，近代的なデジタルコンピューターが考案され，そこで稼働するシステムは経済社会の基盤として必要不可欠の存在になっている。当初は軍事や科学技術計算として機械語やアセンブリ言語といった低水準言語によるプログラム開発が行われたが，より開発の効率を高くする高水準言語の登場により，コンピューターシステムは幅広く一般に活用されるようになった。FORTRAN[3)]（FORmula TRANslation）は最初期の高水準言語であるが，

2） 2009年の時点において世界で2,000億行のCOBOLプログラムが稼働しているとの報告もある（Laird, A. (2009), "The Development of COBOL," *Survey of Programming Languages Spring 2009*, Cedarville University, p. 1）。

3） IBMのJohn Backusが1954年に考案。1957年のIBM704以降に搭載された（Pugh, E. W., Johnson, L. R., Palmer, J. H. (1991), *IBM's 360 and Early 370 Systems*, London: The MIT Press, pp. 38-39）。

図 2-1　配列の出力レイアウト例

```
****** 計算結果 ******
X(99)=99999        Y(99)=99999
   .                  .
   .                  .
   .                  .
```

現代でもスーパーコンピューター用のプログラミング言語として使用されることが多い。FORTRAN は数値計算用の豊富な関数が用意されており，科学技術計算の分野においてその蓄積による利便性は高い。

その後，事務処理系の計算やシステム構築に適する言語として，COBOL が開発された。各種の数値計算に特化した関数主体の FORTRAN に比べて，COBOL は自然言語に近い記述と構文を持ち，より一般的な業務で広く活用されることとなる。COBOL では記述が冗長な分，説明的な言語としてとらえることができることは，開発やその後のメンテナンスにおいて重要な意味を持つ。ここで図 2-1 のようなレイアウトで，１次元配列となっている変数 X と変数 Y の値を出力するコードを考える（“9”は数字１桁分を示す)。これをプログラムで記述するときの様子として，図 2-2 に FORTRAN，図 2-3 に COBOL によるコーディングの一部の例を示す。

図 2-2 において，１行目と３行目は出力ならびにそのフォーマットを意味し，この部分を COBOL で記述しようとすると，図 2-3 のとおり大きく異なるコーディングとなる。図 2-2 の２行目は，N 件の X と Y のデータセットをフォーマットに従って出力する部分である。これに対し COBOL では，図 2-3 のとおりより多くの行を用いてコーディングする必要がある（フォーマットのみで出力のコード部分は省略)。見出し行の定義や明細行の定義は，１行の先頭位置からの文字数に空白や文字，挿入する値の定義を記述する文法となっている。これはプログラマー個人による独自のスタイルを排し，誰の目にも読みやすいという利点といえる。こうした例を見て明らかなとおり，数式に似た記法で書かれる FORTRAN に対して，COBOL の構文は自然言語を意識して冗長な分，解読し易い説明的な書式が採用されていることが分かる。

自然言語に近い COBOL は，今日使用される他の多くのプログラミング言語とは大きく異なった構文となるが，事務処理システムではそれも利点となって1980年代までのメインフレーム全般の時代では，ほとんどのシステムが COBOL で開発されることとなった。事務処理計算は，科学技術計算と異なり単なる数値処理ではなく住所や氏名といった文字データの処理やマスタファイルの入出力操作，そして多くの複雑な帳票を出力することが多い。し

図 2-2 FORTRAN によるコーディング例

```
    WRITE(7, *)  '****** 計算結果 ******'
    WRITE(7, 310)(I, INT(X(I)), I, INT(Y(I)), I=1, N)
310 FORMAT(1H,  'X(',I2,)=' , I5,  '     Y(',I2,  ')=' , I5)
```

図 2-3 COBOL によるコーディング例

```
DATA              DIVISION.
FILE              SECTION.

FD    PRINT FILE
      RECORD CONTAINS 132 CHARACTERS
      LABEL RECORD   IS   OMITTED
      DATA   RECORD   IS   PRINT-REC.
01    PRINT-REC              PIC X(132).
*
WORKING-STORAGE     SECTION.
01  MIDASHIGYO.
    02  FILLER          PIC X(10)   VALUE SPACE.
    02  FILLER          PIC X(22)   VALUE "****** 計算結果 ******".
    02  FILLER          PIC X(100)  VALUE SPACE.
01  MEISAIGYO.
    02  FILLER          PIC X(8)    VALUE SPACE
    02  FILLER          PIC X(2)    VALUE "X(".
    02  X-IND           PIC 9(2).
    02  FILLER          PIC X(2)    VALUE ")=".
    02  X-VAL           PIC 9(5).
    02  FILLER          PIC X(4)    VALUE SPACE.
    02  FILLER          PIC X(2)    VALUE "Y(".
    02  Y-IND           PIC 9(2).
    02  FILLER          PIC X(2)    VALUE ")=".
    02  Y-VAL           PIC 9(5).
    02  FILLER          PIC X(98)   VALUE SPACE.
```

たがって，入力データや出力ファイルのデータ構造，また各種帳票の定義において，その分かり易さが言語の仕様に求められる。この点COBOLは図2-3でも示されるとおり，ひとつひとつの項目を丁寧に記述する構文となっており，コーディングの手間という点では他の言語から圧倒的に不利であるものの，そのドキュメント性からも一旦安定稼働すればメンテナンス時にも威力を発揮する言語であるといえる。そして，良いプログラムがそのまま良いドキュメントになるということは，仕様の意思疎通で大きな問題となるオフショア開発において特に有効な特徴であり，そこにオフショアでの大きな可能性を見出すことができるのである。

1980年代以降，オープンシステムや本格的なオブジェクト指向プログラミングの登場などにより，より生産性の高いプログラミング言語が新規開発では主流となっていった。しかし，既にメインフレーム上での安定した稼働実績を持つ巨大かつ膨大な数のCOBOLシステムは，新規のIT投資をしてはじめから開発し直す動機にはなりにくい点もあった。新規開発でCOBOLによる開発が減った理由として，オブジェクト指向プログラミングの考えやWebをはじめとする他のシステムとの連携機能が，当初のCOBOLにはなかったことが挙げられる。ところが，COBOLは当初の言語仕様であるCOBOL-60から継続して進化を遂げており，現在の仕様ではオブジェクト指向の技術も採用されている。COBOLは事務処理計算用言語としてそれまでの資産との互換性が配慮され，時代に即した課題を取り扱える汎用的な言語として遜色のないものとなっている[4)]。こうした変遷を経て未だ使われ続けるCOBOLは，ライフサイクルの長い言語として突出しているといえる。

2-2　COBOLの規格更新が持つ優位性

BASIC（Beginner's All-purpose Symbolic Instruction Code），PL/I（Programing Language One），Adaといった早くから使われたプログラミング言語がCOBOLを凌駕できなかった点としては，ビジネス処理における利便性はもとより，その規格更新による進化も大きく作用したと考えられる。2002年には，上位互換で第4次となるCOBOLの規格[5)]がCOBOL2002としてISO（the International Organization for Standardization）により制定されている。事務処理計算用の言語として未だ膨大なシステムが稼働するもう1つの理由として，2進化10進数（BCD: Binary-coded decimal）への対応が挙げられる。これは10進法の各桁をBCD（2進数の4桁）で表現するもので，2進数表現においてまるめ誤差が避けら

4)　Schricker, D. (2000), "Cobol for the Next Millennium," *IEEE SOFTWARE*, Vol. 17, No. 2, pp. 48-52.

5)　規格としてはその前に第1次規格（COBOL68），第2次規格（COBOL74），第3次規格（COBOL85）がある。

れない欠点がなくなり正確な計算を実現する方式である。科学技術計算など，他の言語では大きな（あるいは小さな）数値を扱う際に浮動小数点方式を用いるが，倍精度や4倍精度の仕組みはあっても，まるめ誤差の影響を受けることには変わらない。この点で，特に金融システムなど大きな金額や細かい利率計算などが多い事務処理システムでは，効率よく正確な演算を可能にするためには，COBOLが最善の選択となる。

COBOL2002では，オブジェクト指向などプログラミング言語の進化を取り入れた数多くの機能が加えられ，2014年には第5次規格としてCOBOL2008が制定されるに至っている。また日本では国際規格としてJIS X 3002：2011（電子計算機プログラム言語COBOL）として改正された。こうした改定によりたとえばつぎの新機能が実現されている[6)]。

・オブジェクト指向プログラミングがより完成された規格として実現。
　カプセル化や継承という基本的な機能から，クラスとインタフェースの分離，ガベージコレクションの実相などを含んでいる。
・例外処理機能の拡充。
　生産性にも影響する，共通した例外処理機能が適用された。
・国際化対応機能の導入により，変数名として日本語のデータ名が使えるようになり，また日本語データを扱う字類（型の一種）が用意された。

これだけを見ても，プログラミング言語の進化によって登場したC++やJava，あるいはVB（Visual Basic）[7)]といったオブジェクト指向を目指したプログラミング言語と遜色がないことが分かる。さらには，図2-3に示すような説明的なコードは他の言語にはなくCOBOLに限定されるといった優位性がある。

3．レガシーシステムの変遷およびオフショア開発の問題点

3-1　基盤システムとしての変遷

メインフレームやオフコンが主体であった事務処理システムは，1990年代からのオープンシステム化やインターネットの普及と相まって，ダウンサイジングの流れに対峙することになった。これにともない開発用のプログラミング言語は多様化し，教育機関などでのCOBOL教育は影をひそめ，C/C++やJavaほかが多くなっている。クライアント／サーバー型の分散処理方式も進み，従来のシステムはレガシーなシステムともよばれている。またコスト面でもレガシーシステムに対して，安価なオープン系のサーバーシステムが注目を集

6）高木渉（2011）「COBOL–JIS X 3002：2011公示について」（『情報技術標準 NEWSLETTER』No. 89）21ページ。

7）Javaは米国Oracle Corporationの商標登録，VBは米国Microsoft Corporationの商標登録。

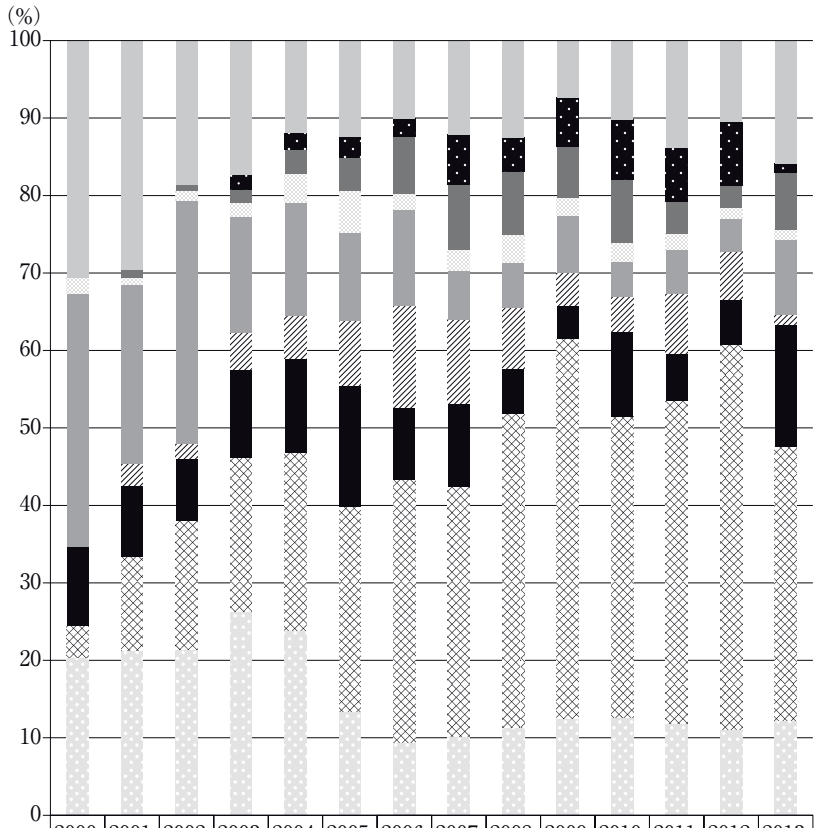
めることとなった。しかしながら，基幹システムを稼働する上での信頼性や安定性，障害時の復旧体制をふまえればコスト面での課題は帳消しとなる。

こうした観点から，安定した高信頼システムの部分はレガシーシステムで，またユーザビリティが要求されるサービスシステムの部分ではオープンシステムで構築，さらには両者を融合させたシステム構築が望ましいことになる。高信頼性安定稼働の中利点の多いCOBOL資産は，新規投資をするリプレース対象とはなりにくい。しかし技術革新によるシステム利用形態の変化は，レガシーシステム単独での運用との間に大きなギャップを生んでいる。これを解決する方策として，COBOL本体のアプリケーション部分はそのまま維持し，システムが構築されているプラットフォームをより有効なものに移行するレガシーマイグレーションが有効になってくる。

情報処理システムの構築の手法は，各種基本設計にはじまる上流工程を経て詳細設計におけるプログラムの設計・開発に進むウォーターフォールモデルや，上流から下流へと部分的に開発を進めていくスパイラルモデルなど，その形態は様々である。しかし最後の段階ではプログラム開発が必ず必要となり，プログラミング言語の選定がシステム設計段階で検討されることになる。ソフトウェア工学の進化は，より利便性が高く効率的な開発を可能とするプログラミング言語の開発に結びついているが，ハードウェア技術の進歩にも大きな影響を受けている。ダウンサイジングによるハードウェア環境は数年で激変し，クラウドに代表されるネットワーク利用やデータベースを経てビッグデータの活用と，そこで用いられるシステムに要求される機能は時代と共に大きな変化を遂げている。したがって，プログラミング言語そのものの開発は，その時代ごとに要求される機能や実現可能性が異なってくるのである。

ここで問題となってくるのが言語の仕様と標準化である。ある言語の仕様が考案された場合，それはハードウェアメーカーによって，たとえばメインフレームであればオペレーティングシステムに言語処理プログラムとして組み込まれることになる。このときに各メーカーがその言語の仕様をどの程度忠実に反映させるか，同じ言語でもメーカーによる差異が出てくることも多い。したがって，時代の要請や技術革新に対応していく規格更新と合わせ，特定のハードウェアに依存しない標準化が重要となってくる。図3-1は新規のシステム開発で用いられたプログラミング言語（または開発環境）件数の経年推移である。COBOLは2000年代以降新規という点では減少しているものの，登場から40年以上経過してもなお一定のシェアを維持している。このことは言語仕様が持つ事務処理計算への適応性はもとより，規格更新ならびに標準化が絶えず行われてきたことを意味するものであるといえる。図3-1において，VBは2000年代前半では非常に注目を集め新規開発も進んだが，2000年代後半には新たな技術導入によりVissual C ++と合わせてVisualBasic. NETへと推移している。1995年

図 3-1 開発言語の変遷

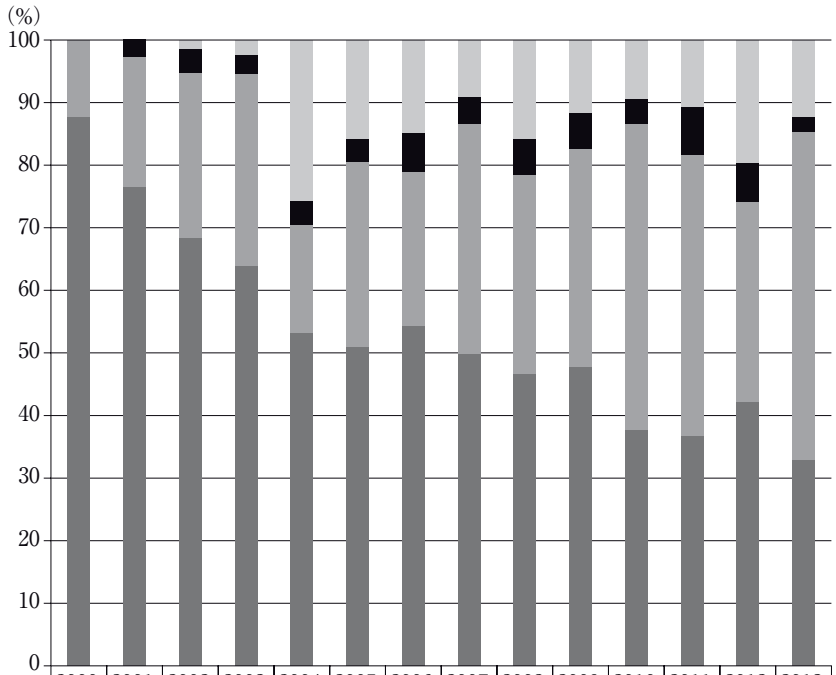


(N=3,029)	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013
その他	15	32	28	29	44	37	30	36	32	19	25	34	22	13
C#				3	8	8	7	19	11	16	19	17	17	1
VisualBasic.NET		1	1	3	11	13	22	25	21	17	20	10	6	6
Visual C++	1	1	2	3	14	16	6	8	9	6	6	5	3	1
VB	16	25	47	25	54	34	37	19	15	19	11	14	9	8
C++		3	3	8	20	25	39	32	20	11	11	19	13	1
C	5	10	12	19	45	47	28	32	15	11	27	15	12	13
Java	2	13	25	33	84	79	101	96	103	126	95	102	104	29
COBOL	10	23	32	44	88	40	28	30	29	32	31	29	23	10

（出所）『ソフトウェア開発データ白書2014-2015』，46ページをもとに作成。

に初めてリリースされた Java は，2000年代後半から急速に使用の頻度が高くなっている。これはゲームコンソールからスーパーコンピューターに至るまで，その動作環境（プラットフォーム）を選ばない利便性や，オブジェクト指向プログラミングによる開発効率の高さによるものといえる。1990年代には主流となってきた C は，システムのカーネル部分の記述

図 3-2　開発プロジェクトの種別



(N=3,050)	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013
拡張			2	4	95	48	44	27	40	30	23	26	41	10
再開発		3	6	5	14	11	19	13	15	15	10	19	13	2
改修・保守	6	23	40	52	64	90	73	110	81	91	121	110	67	43
新規開発	43	85	104	108	197	155	162	149	119	124	93	90	88	27

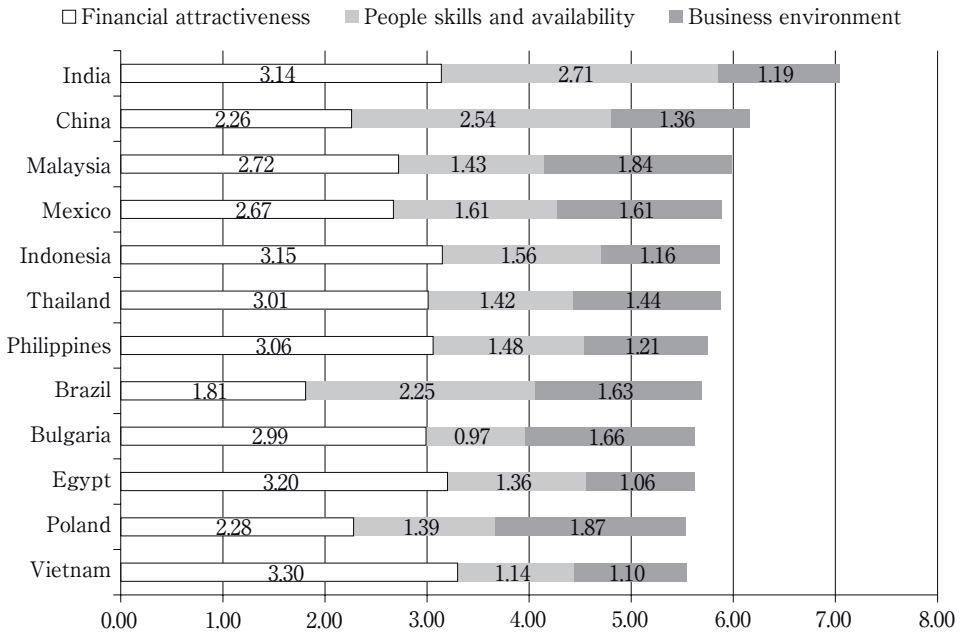
(出所)『ソフトウェア開発データ白書2014-2015』, 36ページをもとに作成。

にも用いられて処理系の基本にある点，またCOBOLと同様にネイティブな言語として実行速度が速いなどの利点もあり，その需要は減っていない。図3-2に，実施された開発プロジェクトの経年推移を示す。新規開発案件は急速に減少し，それに対応する形で改修・保守が増えていることが分かる。

3-2　オフショア開発の現状

オフショア開発は日本も米国も1990年代後半から実施する企業が増えているが，日米ともその目的として開発コスト削減や人材不足の補完が強い動機となっている（ただし，米国では開発のスピードアップを期待する部分も強いのに対して，日本でそれはあまり強い動機とはなっていない)[8)]。米国からは，ちょうど時差で昼夜が逆転し，帰国した留学生とのパイプや英語の語学力，高い技術力が備わるインドへのオフショア開発が著しい。これに対し

図3-3 開発プロジェクトの種別



（出所） Kerney, A. T. (2014), "A Wealth of Choices: From Anywhere on Earth to No Location at All," *The 2014 A. T. Kearney Global Services Location Index*, p. 3.

て，日本からは太い経済のつながりのほか漢字圏という文化や現地日本企業への業務システムサポートという観点から，中国へのオフショア開発が伝統的に多くなっている。こうした日米の違いは，オフショア開発で行うソフトウェアの違いでも明確に棲み分けができていることが分かる。すなわち，米国からは市販を目的としたパッケージソフト開発のウェイトが高く，ソフトウェア自体の技術力ならびにソフトウェア市場での開発スピードにより重点が置かれる。これに対して日本の場合は，社内用の業務アプリケーションの開発が多く，技術やスピードアップは米国ほど顕著に表れてこないのである。米国のオフショア開発はインドそして日本は中国と，非常に高いウェイトで特化しているが，米国よりも比較的オフショア先の相手国数が少なかった日本企業も，2000年代後半からの政治情勢の影響や人件費の高騰もあって，中国に加えて地続きのベトナムへと拡大している。

図 3-3 は，システム開発や運用，バックオフィス業務などのサービス拠点として魅力的な国をランキングしたものである。賃金レベルやインフラコスト，税規制をはじめ，大卒者数

8） 総務省情報通信政策局情報通信経済室（2007）「オフショアリングの進展とその影響に関する調査研究報告書」6－7ページ。

や語学力のほかインフラ整備などから3項目に分けて評価している。トータルスコアの高いほどオフショアとしてポテンシャルが高く，同時に人材の取り合いにもなり易いといえる。ベトナムはトータルスコアでインドや中国に及ばないものの，Financial attractiveness の項目では3.30とスコアが高い。そしてインフラや技術面，語学の点からトータルスコアが低くなっているが，人件費やカントリーリスク，人口の多さや国民性などの点から急速にポテンシャルを増しているものと思われる。

3-3　国内の人材不足とオフショア開発がかかえる問題

慢性的な技術者不足ならびに開発が先延ばしになっているバックログの問題をかけるソフトウェア開発において，1990年代から国外の拠点や開発ベンダーに開発を委託するオフショア開発がはじまった。背景には国内の人材不足に加えて人件費のコスト削減があったが，オフショア開発先として挙げられる中国やインドでは，国家政策による高度ICT人材の育成に伴い人件費からの魅力は減少している。これに対して新たな市場も注目され，ベトナムさらにはミャンマーなどへの人材囲い込みに発展してきている。

ICT人材市場が日本や欧米から日本を除くアジアに重点がシフトする中，情報工学系に限らず大学卒業者数は依然として伸びが高く，こうしたアジア諸国への依存傾向は変わらない。しかし，限られた条件のもと高度ICT人材の育成や人材の取り合いによる人件費への圧迫は避けられず，ソフトウェア開発の上流工程を国内で，プログラミングを主体とする基礎的な人材供給をオフショアに求める戦略には転換が必要である。すなわち，オフショアベンダーを多重下請の1つとして下流工程をまかせるのではなく，長期的なビジネスパートナーとして対等な関係を構築し，一体化されたプロジェクトチームを作っていくことが重要といえる。一方で，高度なスキルを達成したエンジニアはキャリアパスの中でより給与の高いポジションへと移動することが顕著であり，進出企業からすると新規開拓をして人材育成に力を入れても離職率は高い，というジレンマをかかえている。こうした中で日本企業のオフショア開発戦略は，ブリッジエンジニアの奮闘や日本国内でのオンショア開発導入であっても解決されない課題をかかえているのである。

人材不足や国内経済の影響は，新規開発案件の凍結にも影響した。2000年代に入り有効求人倍率の平均は2006年の1.06をピークに2009年には0.48と減少したものの，2014年には1.09と2009年を超える水準になった（総務省『一般職業紹介状況（長期時系列表)』より算出)。そして，この間に凍結されていた大規模なソフトウェア案件や2016年に開始される「マイナンバー社会保障・税番号制度」のシステム導入など，数千億円規模の開発プロジェクトの始動が想定されている。しかし「2015年問題」として，IT受託案件がこの数年で集中するもののその後に続く大規模需要が想定されていない問題が指摘されている。ソフトウェア業界

の求人数は大きく伸びているものの，各企業は東京五輪までの数年間というピークを過ぎた後の人件費圧迫に対応する必要があるのである。ここに日本のソフトウェア業界で行われてきた多重下請によるコスト削減の慣習と相まって，オフショア委託においても一時しのぎの圧迫から2000年代以前の開発スタイルに戻るのではないかという，もう一つの2015年問題を抱えているということができる。

もともと規模が大きくなる金融系システムでは，メガバンクへの再編と相まって問題をかかえながらもシステムの統合や新規開発が進み，今後それらが一段落すれば運用およびメンテナンスのフェーズに入ることになる。メガバンクによる基幹システムの一本化は数千億円規模の IT 投資となっているが，三菱東京 UFJ 銀行や三井住友銀行は勘定系システムの刷新を終え，みずほ銀行もこの数年で終了すれば，巨大な COBOL 資産をかかえる金融系システムでは今後長期に亘りメンテナンスのフェーズに入ることになる。他の業務システムについても，民間や企業の区別なく基幹業務の開発が一巡すれば安定期に入るものと思われる。バックログをかかえるも，既定の大規模案件ほどには積極的な開発投資は行えず，コスト減を期待したオフショア開発へと目が向くのは明らかである。

オフショアの日米比較をとらえたとき，米国が上流工程から下流工程まで協業体制で進めているのに対し，日本ではプログラム開発や単体テストといった下流工程が主であり，伝統的な多重下請業態がそのままオフショアでも適用されている。さらに仕様書や設計書の詰めが甘く，開発しながらの仕様変更など国内で悪しき習慣となっていた開発スタイルも引き継がれている傾向にある。こうした点は，チームとして協業体制を行っていく上での大きな障害であり，日本企業への人材の定着にも難点があるといえる。上流工程も含んだプロジェクトでは，プロジェクトマネージメントはもとより高度なスキルが要求され，収入面やキャリアパスという観点からも大きな関心点となる。そこをとらえているのが米国であり，次々と人件費の安い国を開拓していく日本のやり方では，限界に行き着くものと思われる。

4．レガシーマイグレーションに適用可能なオフショア開発

4-1 レガシーシステムのメンテナンス問題

1947年前後の団塊世代が60歳定年を迎える2007年は，2007年問題として注目を集めたが，コンピューター業界では特別に大きな時点としてとらえられた。1970年代から80年代に至る，メインフレームによる COBOL システムの開発が全盛期を迎えていた時代にエンジニアだったのが，この団塊の世代だからである。90年代以降，新たに COBOL を学んで就職してくる学生はいなくなり，開発業務でも新規案件については COBOL 以外の言語で既に要員不足の状態となった。したがって COBOL プログラミングのスキルがそれまでのように次の世代に引き継がれず，COBOL 資産のメンテナスや新規開発で団塊の世代を中心とす

る層に頼らざるをえない状況となってしまった。

しかも，COBOLシステムをそのまま使うことが時代の流れとともに困難になり，プラットフォームを変えたオープン化への移行が重要になってきた。すなわち，COBOL資産のメンテナスや新規開発では，従来からのCOBOLのスキルのほか，新たに仕様として付け加えられたオブジェクト指向をはじめとする機能を合わせて熟知する必要性あるのである。さらにプラットフォームを変え，オープンシステム上でこれまでのメインフレームを用いた端末処理は，クラウドも用いたWebブラウザを用いた処理へと変更される。そこで業務システムの本体であるCOBOLの部分とのデータのやりとりで，Javaをはじめとする別の言語で構築しなければならなくなった。より進化が進むCOBOLに加え，常に時代に即した高い技術が求められるようになった訳である。ただでさえ技術者が不足する中，COBOLでのオープンシステム化という高度なスキルを持つ多くの技術者を確保するのが難しく，開発や移行を延期した案件も数多く出ている。

基幹業務以外で比較的規模の限られたCOBOLシステムでは，Javaなどですべてを置き換える新規開発も進んでいる。しかし安定性と信頼性を要求する銀行の勘定系システムなどでは，それは困難なことである。COBOL以外で今日多く用いられるプログラミング言語には，1990年代以降にようやく本格的に業務で使われ出したものが多い。そこで言語の仕様や安全性，また構築されたシステムの安定性などの点で，重要な基幹業務に適したものであるというレベルには達していないのである。またメインフレーム以外のプラットフォームで，止めないシステムとして連続運転できる保証もない。障害時には完全なログにより，トランザクションの詳しい分析が行われるが，そうした対応がCOBOLのシステム以外でも同様に可能であるかなど，簡単にすべてを移行できるという話にはならない。

メインフレームは今日なお生産と出荷が続き，減少傾向にあるとはいえ，大きな需要があることは表4-1のとおり明確である。表に示されたIAサーバーの金額は2004年以降大きな額となっている。しかし台数自体が非常に多いことから金額が大きくなる側面があり，PCアーキテクチャによる個々のサーバーとしてとらえると，メインフレームへの投資が今なお本格的なものであることが分かる。なおメインフレームでは，COBOL資産をそのままプラットフォームを移すことでコスト削減とレガシーシステムの新技術への対応を図るケースも多くある。そこで表4-1におけるメインフレームの台数減少は，単純にCOBOLシステムの減少ととらえることはできない。

表 4-1 メインフレームの出荷推移

（単位：台（上段），百万円（下段））

	1998	1999	2000	2001	2002	2003	2004	2005
メインフレーム	3,034	2,059	1,490	1,574	1,305	1,241	1,212	949
	745,685	591,637	499,068	474,540	370,151	262,539	246,721	193,334
UNIX サーバー	－	－	－	－	－	－	59,666	62,735
	－	－	－	－	－	－	334,805	322,803
IA サーバー	－	－	－	－	－	－	368,564	414,283
	－	－	－	－	－	－	277,650	300,672
ワークステーション	－	－	－	－	－	－	100,278	151,109
	－	－	－	－	－	－	48,202	530,020

	2006	2007	2008	2009	2010	2011	2012	2013
メインフレーム	872	713	601	562	450	390	394	341
	180,117	165,836	119,694	118,600	82,563	60,279	80,306	71,408
UNIX サーバー	59,161	42,818	29,922	21,285	13,725	10,699	8,590	1,849
	280,068	223,762	176,772	122,531	111,273	101,130	81,318	7,516
IA サーバー	449,708	326,718	317,132	313,097	314,259	332,242	321,678	340,889
	308,826	256,130	231,905	194,362	192,890	196,541	198,655	211,021
ワークステーション	146,746	105,214	96,617	95,165	73,665	73,340	75,932	74,408
	51,112	28,627	25,180	20,320	12,118	12,055	12,925	13,379

（出所） 社団法人電子情報技術産業協会『統計データ／サーバ・ワークステーション』をもとに作成。

4-2 オープン系システムへのマイグレーション

日本ではメインフレームのメーカーが集まる世界有数の国となっている。米国 IBM を頂点にした数社を加えれば，ほぼすべての世界出荷が賄われていることになる。それだけに，COBOL 資産とその技術の蓄積は米国を除く国々とは群を抜いて高度な状況にある。特にオフショア先の国々では，メインフレームの時代を体験せずにオープン化へと社会が進んでいる。ここで日本の現状をとらえると，つぎのとおり問題点を挙げることができる。

・ソフトウェア開発要員の慢性的な不足
・COBOL 技術者の先細り
・COBOL 資産が膨大である
・基幹業務では膨大な COBOL 資産を維持する必要がある
・オープン化によるコスト削減が求められる
・クラウドをはじめとした新技術への対応が必要
・オフショア開発では高度 ICT 人材は離職しがちである

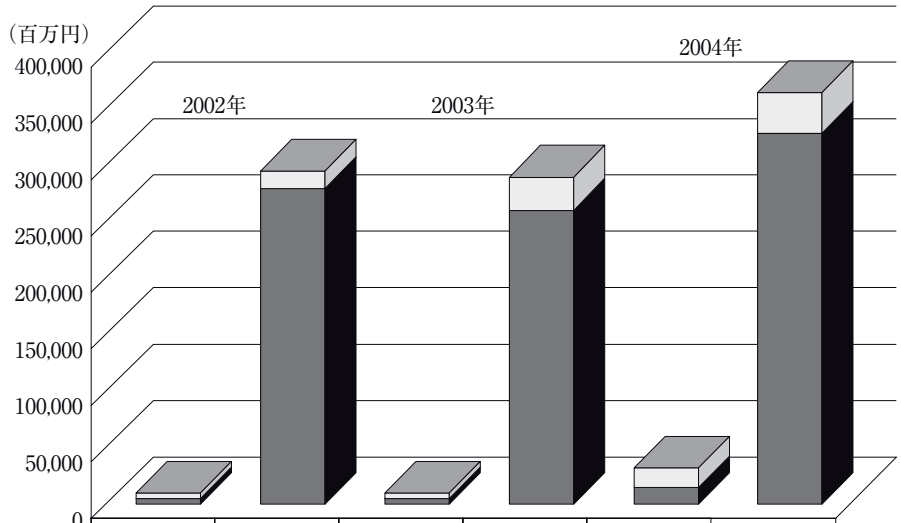
・日本の慣習としてそのまま下流工程の下請けにするオフショアは成り立たなくなる

これに対して，オフショアの相手国ではつぎの点が今後考慮されるべきである。まず経済社会の発展とともに，基幹業務の自国開発が必要となってくる。そこでは豊富な知識と技術に基づいて業務の分析や設計を行っていくことになるが，経験なしにはプロジェクトの管理からして不可能である。そうした中，対応できる高度な技術者を育成するには，日本のCOBOLシステムの資産移行に携わることが非常に有効となる。不特定のユーザーを対象とした，いわゆるパッケージソフトの開発が主な米国型オフショアと異なり，日本からのオフショアは内部業務用のアプリケーションを豊富に扱ってきた。こうした側面から考えると，基幹業務のための高度な技術者育成には日本だけが貢献できる状況にあることが分かる。幸い，COBOLの仕様はより自然言語に近く，ソースコードはドキュメントに準じてとらえることができる。さらにJavaや他の言語に比べて書法は厳格な側面があり，プログラマー個人の資質や考え方によって書き方が変わるということが少ない。この点はプログラマーからみれば面倒な制約になるが，作成者によらず品質を維持したプログラム開発を達成する上での重要な仕組みとなっている。以上のことからも，COBOLはオフショアに非常に適したプログラミング言語でもあると考えることができる。

図4-1は，日本のソフトウェア輸出入実績（PC用ゲームソフトを除く）である。どの年も圧倒的な輸入超過であり，とくに米国からのソフトウェア輸入は2002年で全体の95%，2003年および2004年では90%を占めている。輸出に比べて僅かな額の輸出においては，多くの場合は海外の現地日本企業向けのアプリケーション開発であり，市販パッケージソフトの輸出はほとんど見られない。こうした傾向は，2010年代も同様に続いており，日本のソフトウェア産業の活路は未だ見出せていないのが現状である。

しかしながらオフショアを行う日本の企業にとって，これまで示してきた日本国内の状況は非常に有利であるということもできる。オフショアで人材育成をして，技術者として成長したら欧米の企業へ移ってしまったというケースは非常に多い。しかし上記で述べたオフショアマイグレーションの観点から取り組んだ場合，離職の行き先はスキルを活かせる日本のベンダーが筆頭候補となり，全体として人材の流出にはならない。優れた実地教材としてCOBOL資産に携わりながら最新の技術仕様を習得し，さらにはオープンシステムへ移行のためにプラットフォームへの資産移行やCOBOLと他のシステム間の通信など，より高度な技術の習得を実現するものとしてオフショア相手国への貢献度は大きい。日本国内でも人員不足が解消してオープン化へのスムーズな移行が実現することになる。さらにはオフショア先での基幹業務構築が本格化した際，ハードウェアはもとより，逆のオフショアとして日本でのシステム開発も十分に発生すると思われる。そして，その次に訪れるステップは，オフショア先相手国基幹業務向けのソフトウェア輸出にほかならない。

図 4-1 ソフトウェア輸出入実績



	輸 出	輸 入	輸 出	輸 入	輸 出	輸 入
□米国以外	4,184	15,015	4,108	29,485	17,525	35,383
■米　国	5,131	279,653	5,105	260,090	14,466	329,200

（出所） 社団法人情報サービス産業協会，社団法人電子情報技術産業協会，社団法人日本パーソナルコンピュータソフトウェア協会『2005年コンピュータソフトウェア分野における海外取引および外国人就労に関する実態調査』（2005）をもとに作成。2002年から2003年にかけての，それぞれ262社，251社，318社の調査による。

5．おわりに

メインフレームおよびそこで構築されてきたCOBOL資産はシステムとして安定稼働し，高い信頼性が必要とされる基幹業務で不可欠の基盤となっている。また技術的にも50年以上の歴史を持つCOBOLは，既に安定に入りながらもオブジェクト指向の概念など多くの新技術を取り入れて進化を続けている。それにより，基幹業務の処理本体がビジネスロジックとして記述されているCOBOLでの手続きがカプセル化され，本体部分をいじらず他の言語から利用可能な機能も提供できることとなった。またプラットフォームを選ばず，オープンシステムとしてメインフレームからのダウンサイジングが実現されている。こうしたことは，Javaなどのプログラムから COBOL プログラムを直接呼出す連携，また各種の Web サービスにおいて，信頼性の中核として価値を持つ COBOL 資産活用の広がりを示すものといえる。

オフショア開発がかかえる課題について，COBOL資産の特質をふまえながら解決する有効な方法を示した。オフショア委託先の国々の基盤にはないメインフレームやCOBOLシステム，また基幹業務ロジックの文化に関しては，情報として流布されることが少ない。ま

た国内でも新規開発という観点からは，COBOL以外のプログラミング言語に絶えず注目が集まっている。こうした点は，大学や専門学校での教育においてCOBOLの学習がほとんど見られなくなり，当然の結果としてCOBOL資産の価値や意義を理解するべき部分の欠如が，一部を除いて社会全体に広がっていることを意味する。これは日本のソフトウェア産業がかかえる大きな盲点である。しかし世界でも有数のメインフレーム大国である日本は，オフショアの活用や圧倒的な輸入超過となっている伝統的な産業構造変革の意味で，大きなチャンスを持っているといえる。情報専門学科で設置されているカリキュラム標準などでは，本論で指摘したCOBOL資産を用いたオープンシステムをオフショアマイグレーションとしてとらえる知見が含まれていない。これは銀行の勘定系業務など，社会の基盤となる経済社会の学問分野からの考慮がなく，エンジニアリングだけの視点では不足であることを意味する。そして，ソフトウェア産業の活路を見出すためには，経済系学部における情報カリキュラム標準を設けた上での人材育成が望まれることになる。

参考文献

社団法人情報サービス産業協会，社団法人電子情報技術産業協会，社団法人日本パーソナルコンピュータソフトウェア協会（2005）『2005年コンピュータソフトウェア分野における海外取引および外国人就労に関する実態調査』。

社団法人電子情報技術産業協会『統計データ／サーバ・ワークステーション』。

高木渉（2011）「COBOL－JIS X 3002：2011公示について」（『情報技術標準 NEWSLETTER』No.89）21ページ。

独立行政法人情報処理推進機構技術本部ソフトウェア高信頼化センター（2014）『ソフトウェア開発データ白書2014-2015』IPA，36，46ページ。

Bermer, R. W. (1971), “a view of The History of COBOL,” *Honeywell Computer Journal*, Vol. 5, No. 3, pp. 132-133.

Kerney, A. T. (2014), “A Wealth of Choices: FromAnywhere on Earth to No Location at All,” *The 2014A. T. Kearney Global Services Location Index*, p. 3.

Laird, A. (2009), “The Development of COBOL,” *Survey of Programming Languages Spring 2009*, Cedarville University, p. 1.

Pugh, E. W., Johnson, L. R., Palmer, J. H. (1991), *IBM’s 360 and Early 370 Systems*, London: The MIT Press, pp. 38-39.

Schricker, D. (2000), “Cobol for the Next Millennium,” *IEEE SOFTWARE*, Vol. 17, No. 2, pp. 48-52.